Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Črt Gregorič

# Razvoj mobilnega robota vodenega preko iOS aplikacije

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana 2015



**Izvorna koda robota je dostopna na naslovu:**

*https://github.com/crtgregoric/RPi-Robot.git*

**Izvorna koda mobilne aplikacije je dostopna na naslovu:**

*https://github.com/crtgregoric/RPi-Robot-Control.git*

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi celovito predstavite razvoj daljinsko vodenega mobilnega robota od ideje do delujočega izdelka, pri čemer naj bo glavno vodilo čimbolj funkcionalna rešitev za najboljšo ceno. Robota naj bo mogoče z daljinskim upravljanjem tekoče voziti po nezahtevnih površinah, vsebuje pa naj tudi premično kamero in ustrezno osvetlitev. Strojni del rešitve naj bo zasnovan na računalniku Raspberry Pi in ustrezni kameri, poskrbeti pa morate tudi za robustni način premikanja. Programski del rešitve naj sestavlja uporabniku prijazna mobilna aplikacija na platformi iOS, ki preko internetne povezave upravlja premikanje robota in v realnem času prikazuje pretočni video z robotove kamere, ter program v Pythonu, ki ima vlogo robotovih možganov.

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Črt Gregorič, z vpisno številko **63110045**, sem avtor diplomskega dela z naslovom:

*Razvoj mobilnega robota vodenega preko iOS aplikacije*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 10. februarja 2015 Podpis avtorja:

Posvečeno Duškotu, Zdenki, Roku, Nevi in malemu Jakobu.

# Kazalo

**Povzetek**

**Abstract**

**1 Uvod** **1**

**2 Razvoj mobilnega robota vodenega preko iOS aplikacije** **3**

2.1 Načrtovanje . . . . . 3

2.2 Strojna oprema . . . . . 5

2.2.1 Raspberry Pi . . . . . 5

2.2.2 Raspberry Pi Camera Module . . . . . 7

2.2.3 Adafruit PWM/Servo Driver . . . . . 8

2.2.4 Servo motorji in svetleče diode . . . . . 10

2.2.5 Kartica SD . . . . . 10

2.2.6 Sprejemnik WiFi . . . . . 11

2.2.7 Baterije . . . . . 12

2.3 Programska oprema . . . . . 14

2.3.1 Occidentalis . . . . . 15

2.3.2 GStreamer . . . . . 15

2.4 Tehnologije . . . . . 16

2.4.1 Pulzno-širinska modulacija . . . . . 16

2.4.2 Diferencialni krmilni sistem . . . . . 18

2.5 Razvoj . . . . . 20

2.5.1 Priprava . . . . . 20

2.5.1.1 Priprava kartice SD . . . . . 20
2.5.1.2 Povezava naprav PWM . . . . . 22
2.5.2 Programiranje . . . . . 22
2.5.2.1 Upravljanje naprav PWM . . . . . 24
2.5.2.2 Mobilna aplikacija . . . . . 26
2.5.2.3 Sporazumevanje odjemalca s strežnikom . . . 29
2.5.2.4 Video prenos . . . . . 34
2.5.3 Združevanje v celoto . . . . . 39
2.5.3.1 Ohišje . . . . . 39
2.5.3.2 Samodejni zagon . . . . . 40
2.5.3.3 Zaustavitev sistema . . . . . 40

**3 Sklepne ugotovitve** **45**

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **LED** | light-emitting diode | svetleča dioda |
| **USB** | universal serial bus | univerzalno serijsko vodilo |
| **GPIO** | general-purpose input/output | splošno namenski vhod/izhod |
| **LCD** | liquid-crystal display | zaslon s tekočimi kristali |
| **CSI** | camera serial interface | zaporedni vmesnik kamere |
| **FPS** | frames per second | sličic na sekundo |
| **VGA** | video graphics array | grafično video polje |
| **PWM** | pulse-width modulation | pulzno-širinska modulacija |
| **I2C** | inter-integrated circuit | protokol komunikacije med vezji |
| **UBEC** | universal battery eliminator circuit | linearni stabilizator napetosti |
| **SDK** | software development kit | paket razvoja programske opreme |
| **SSH** | secure shell | varna lupina |
| **SSHFS** | SSH file system | datotečni sistem SSH |

# Povzetek

Diplomsko delo predstavlja proces načrtovanja in razvoja daljinsko vodenega mobilnega robota in programske opreme, namenjene njegovemu upravljanju. Poleg razvoja robota in potrebne programske opreme je cilj diplomskega dela tudi preveriti, kako se obnese uporaba računalnika Raspberry Pi pri projektih, povezanih z robotiko. Z uporabo po meri razvite mobilne aplikacije za platformo iOS je mogoče robota voditi po prostoru, spremljati pretočni video njegove video kamere ter spreminjati naklon njenega vidnega polja. Uporaba petih svetlečih diod omogoča spremljanje video prenosa tudi v slabo osvetljenih prostorih. Mobilna aplikacija komunicira preko internetnega omrežja s po meri razvitim programom, napisanim v programskem jeziku Python, ki upravlja delovanje robota in prenosa pretočnega videa. Naloga opisuje postopek načrtovanja robota, postopek izbire uporabljene strojne in programske opreme, opis uporabljenih tehnologij in razvoja lastne programske opreme ter postopek združevanja robota v zaključeno celoto.

**Ključne besede:** mobilni robot, Raspberry Pi, mobilna aplikacija, iOS, pretočni video.

# Abstract

This thesis presents the design process and development of a mobile robot and all the necessary software to control it. The goal of the thesis is also to determine, whether a Raspberry Pi computer is a suitable platform to use for robotics related projects. The iOS application provides the ability to guide the robot, watch its video stream and control the angle of its video camera. By using five LED diodes for illumination it is possible to watch the video stream in low light conditions. The mobile application communicates over the network with a program written in Python, that runs on the Raspberry Pi and manages all robot's functionalities. This work describes the design process of a robot, appropriate hardware and software selection, software development and integration of all needed components into a finished product.

**Keywords:** mobile robot, Raspberry Pi, mobile application, iOS, video stream.

# Poglavje 1

# Uvod

Diplomska naloga opisuje proces razvoja mobilnega robota in programske opreme za njegovo uporabo. Tema je bila izbrana zaradi želje po razvoju lastnega daljinsko vodenega robota, ki temelji na računalniku Raspberry Pi in omogoča spremljanje pretočnega videa v realnem času. Računalnik Raspberry Pi je cenovno zelo dostopen in omogoča zelo raznoliko uporabo. Prav zaradi tega je zelo priljubljeno orodje mnogih hobi navdušencev. Izbran je bil z namenom boljšega spoznanja njegovega delovanja in širine njegove uporabe. Namen diplomske naloge je tudi ugotoviti, ali je računalnik Raspberry Pi primerna platforma za razvoj mobilnega robota.

Z uporabo po meri razvite mobilne aplikacije za platformo iOS bo razvitega robota mogoče nadzorovati in upravljati. Robot bo opremljen z video kamero, preko katere bo omogočal spremljanje pretočnega videa v realnem času. Tako bo robota mogoče upravljati tudi izven vidnega polja upravljalca. Z uporabo svetlečih diod (diod LED) bo nudil možnost osvetlitve vidnega polja kamere. Za boljši pregled nad prostorom in ovirami bo z uporabo dodatnega servo motorja možno spreminjati tudi naklon kamere.

Programska oprema, ki bo omogočala upravljanje robota, se bo delila na dva dela. Prvi del programske opreme bo predstavljala mobilna aplikacija, ki bo opravljala vlogo daljinskega upravljalnika robota. Preko elementov na uporabniškem vmesniku bo uporabnik lahko robota vodil po prostoru,

spreminjal naklon kamere, nastavljal svetilnost svetlečih diod in spremljal pretočni video kamere robota. Drugi del razvite programske opreme pa bo program, napisan v programskem jeziku Python, ki bo opravljal vlogo možganov robota. Skrbel bo za komunikacijo z mobilno aplikacijo in glede na prejete ukaze izvedel ustrezno akcijo. Njegova naloga bo tudi upravljanje prenosa pretočnega videa.

Jedro diplomskega dela je razdeljeno na več poglavij. Vsako opisuje posamezno področje razvoja robota in programske opreme. Začne se z opisom načrtovanja robota. To poglavje opisuje proces opredelitve potrebnih komponent in raziskavo njihove medsebojne združljivosti. Sledi opis uporabljene strojne in programske opreme. Ti dve podpoglavji vsebujeta opis uporabljene strojne in programske opreme ter razloge za njihovo izbiro. Poglavje s strojno opremo vsebuje tudi nekaj opozoril, na kaj vse je potrebno biti pozoren pri izbiri določene komponente. V naslednjem poglavju so opisi uporabljenih tehnologij ter razlage njihovega delovanja in namena. Poglavje ki vsebuje razvoj robota je prav tako razdeljeno na več podpoglavij. Začne se z opisom vzpostavitve računalnika Raspberry Pi in povezave PWM naprav. Sledijo poglavja z opisi razvoja lastne knjižnice za delo z napravami PWM, razvoja mobilne aplikacije in njenega uporabniškega vmesnika, opisi izvedbe komunikacije med robotom in mobilno aplikacijo ter prenosa in prikazovanja pretočnega videa robota. Zakjuči se z opisom izdelave ohišja robota in izvedbo nekaterih dodatnih funkcij, ki služijo doseganju boljše uporabniške izkušnje.

# Poglavje 2

# Razvoj mobilnega robota vodenega preko iOS aplikacije

## 2.1 Načrtovanje

Prvi korak razvoja robota je bilo njegovo temeljito načrtovanje. Ta korak je zelo pomemben, saj si ob načrtovanju ustvarimo širšo sliko o robotu in o komponentah, ki jih pri razvoju potrebujemo. Dobro načrtovanje omogoča bolj učinkovit razvoj, saj lahko morebitne težave prej odkrijemo in se jim ustrezno izognemo.

Preden lahko izberemo ustrezne komponente in začrtamo njihovo postavitev na robotu, je potrebno najprej opredeliti vsa opravila, ki jih bo robot znal izvajati oziroma določiti, kakšno funkcijo bo robot sploh opravljal. Ob tem koraku sestavimo seznam vseh komponent, ki jih potrebujemo za razvoj robota.

V osnovi bo izdelani robot preko posebej razvite mobilne aplikacije omogočal vodenje po prostoru in spremljanje pretočnega videa kamere. Da bi bilo vodenje robota mogoče tudi v prostorih s slabo razsvetljavo oziroma celo v temi, bo robot imel možnost osvetlitve vidnega polja kamere z uporabo svetlečih diod. Prav tako bo z uporabo dodatnega servo motorja mogoče spreminjati tudi naklon video kamere za lažje pregledovanje prostora.

Da bo robot lahko uspešno izvajal zahtevane naloge, bomo za njegovo

izdelavo potrebovali video kamero, pogonske motorje, servo motor za spreminjanje naklona video kamere, nekaj svetlečih diod in razvojno ploščico, ki bo skrbela za komunikacijo z mobilno aplikacijo ter upravljala delovanje motorjev, kamere in svetlečih diod. Ker bo robot daljinsko voden, bomo potrebovali tudi način brezžične povezave. Za dosego popolne prostorske svobode robota bo potrebna uporaba ustrezne baterije, ki bo robota napajala z električno energijo.

Ko imamo spisek potrebnih komponent za realizacijo robota, sledi proces raziskave. V tem koraku raziščemo, katere komponente so najbolj ustrezne za uporabo, preverimo njihovo medsebojno združljivost [1] in morebiti odkrijemo potrebo po uporabi dodatnih komponent, ki jih v procesu opredelitve nalog robota nismo predvideli. Zaradi zastavljenih ciljev diplomskega dela, bo izdelani robot temeljil na uporabi računalnika Raspberry Pi [2]. Dve najbolj pomembni nalogi robota sta njegovo vodenje in prenašanje pretočnega videa, zato sledi raziskava možnosti za izvedbo teh dveh sposobnosti.

Vodenje robota omogoča uporaba servo motorjev. Prvi korak pri raziskavi je bil pregled možnosti upravljanja servo motorjev z uporabo računalnika Raspberry Pi. Ta se je kmalu izkazal kot nezmožen opravljanja zahtevane naloge. Razlogi so bolje opisani v poglavju 2.2.3. Kot rešitev se je pojavila uporaba razširitvenega vezja Adafruit PWM/Servo Driver [3], ki omogoča računalniku Raspberry Pi neodvisno nadzorovanje do šestnajstih naprav PWM. V povezavi z nalogo vodenja robota je sledila raziskava najbolj primernih servo motorjev. Pomemben faktor pri izbiri motorjev je bilo razmerje med ceno, težo, močjo in velikostjo servo motorja. Raziskava je pripeljala do ugotovitve, da so pogonski motorji (to so motorji ki omogočajo neprekinjeno vrtenje ročice servo motorja) več kot dvakrat dražji od običajnih servo motorjev, pri katerih je kot delovanja ročice običajno omejen na 180°. Poleg tega pa je bilo ugotovljeno, da je mogoče običajen servo motor na zelo preprost način predelati tako, da omogoča neprekinjeno vrtenje ročice. Zaradi tega sta bila za pogon izbrana dva običajna servo motorja, na katerih je bila izvedena predelava. Poleg servo motorjev za pogon robota potrebu-

jemo tudi servo motor za nastavljanje naklona video kamere. Ta naloga ne zahteva uporabe zmogljivega motorja z neprekinjenim vrtenjem ročice, zato je bil uporabljen običajen servo motor manjše velikosti in zmogljivosti.

Druga pomembna naloga robota je prenašanje pretočnega videa. Za opravljanje te naloge potrebujemo video kamero, ki je združljiva z računalnikom Raspberry Pi. Raziskava je razkrila množico delno oziroma popolnoma nezdružljivih video kamer. Večina združljivih kamer ni prišla v poštev zaradi visokih cen ali neprimerne velikosti. Zaradi zelo majhne velikosti, relativno nizke cene in zagotovljene združljivosti se je kot najboljša izbira izkazala video kamera, ki je bila razvita namenoma za računalnik Raspberry Pi.

Sledila je raziskava najbolj primernega načina komunikacije. Izbrana je bila komunikacija preko internetnega omrežja. To je zahtevalo uporabo sprejemnika WiFi. Raziskava je zajemala pregled z računalnikom Raspberry Pi združljivih sprejemnikov. Glavna dejavnika pri izbiri sprejemnika sta bila poleg združljivosti tudi cena in velikost.

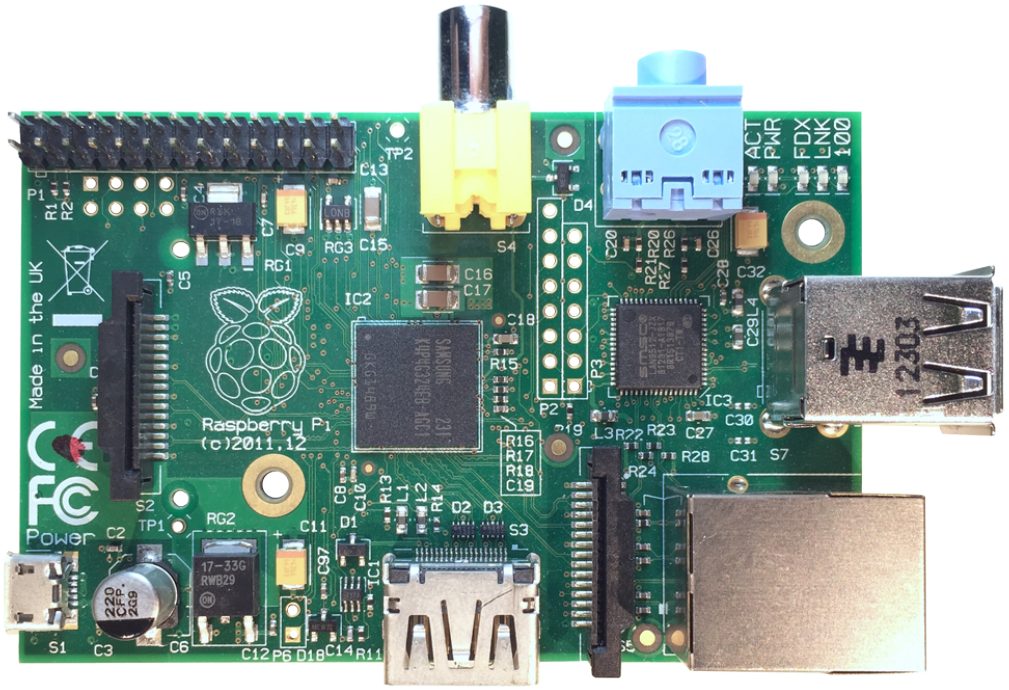
Zadnji del raziskave je zajemal pregled najbolj ustreznih virov napajanja računalnika Raspberry Pi in naprav PWM. Rezultati so bolje predstavljeni v poglavju 2.2.7.

## 2.2 Strojna oprema

Robot je skupek strojne in programske opreme, ki ob sodelovanju omogočata robotu opravljanje in reševanje različnih opravil. Izbira ustrezne strojne opreme je pri razvoju robota kjučnega pomena, saj je osnova, na kateri se razvija programska oprema. Ustrezno izbrana strojna oprema zagotavlja dobro mehansko delovanje robota in v večini primerov olajša razvoj programske opreme. Pri tem je potrebno biti zelo pozoren na to, da so izbrane komponente med seboj združljive.

### 2.2.1 Raspberry Pi

Raspberry Pi [2] je miniaturni računalnik (velikosti kreditne kartice) razvit s strani angleške fundacije Raspberry Pi [4]. Njegov primarni namen je




Slika 2.1: Računalnik Raspberry Pi.

širjenje znanja računalništva in programiranja med otroci, vendar je zaradi nizke cene, nizke porabe električne energije in vsestranske uporabe postal zelo priljubljena platforma za razvoj raznih hobi projektov.

Kot pomnilniško napravo uporablja kartico SD, na katero je potrebno predhodno naložiti operacijski sistem. Ker procesna enota temelji na arhitekturi ARM, lahko poganja samo operacijske sisteme prilagojene tej arhitekturi. Veliko število operacijskih sistemov prilagojenih računalniku Raspberry Pi temelji na jedru operacijskega sistema Linux.

Računalnik Raspberry Pi poznamo v petih različicah: model A, B, A+, B+ in B2. Modeli se med seboj razlikujejo po porabi električne energije, številu perifernih enot, velikosti in zmogljivosti. Za razvoj robota je bil izbran računalnik Raspberry Pi modela B (slika 2.1), predvsem zaradi razmerja med močjo procesne enote in porabo električne energije, svoje velikosti in zaradi perifernih enot, ki so omogočile lažje testiranje in stopnjevanje razvoja

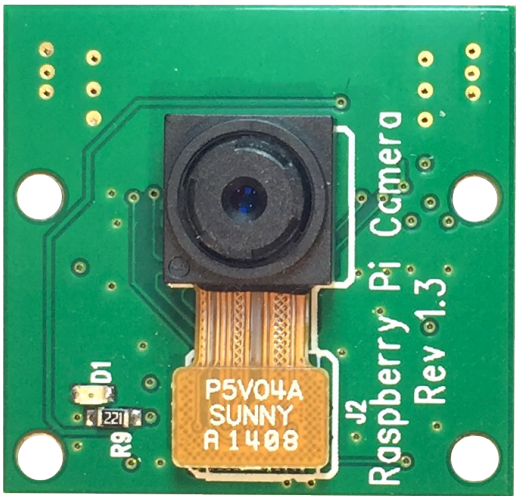
robota. Uporaba modela A bi bila zaradi nižje porabe električne energije in nekoliko manjše velikosti bolj ustrezna za končni produkt, vendar bi zaradi pomanjkanja perifernih enot otežila testiranje in razvoj v začetnih fazah. Glavne specifikacije modela B:

- ethernet priključek
- dvojni USB priključek
- procesna enota ARM11 (ARMv6) s frekvenco 700 MHz
- 512 MB bralno-pisalnega pomnilnika
- napajanje 5 V @ 700 mA
- HDMI prikjuček

Poleg perifernih enot ima tudi 26 (40 pri modelih A+ in B+) splošno namenskih vhodno-izhodnih (GPIO) nožic namenjenih napajanju in povezovanju drugih naprav preko različnih vmesnikov. Nožice GPIO so ključnega pomena pri mnogih projektih, saj omogočajo povezovanje naprav kot so servo motorji, svetleče diode, razširitvena vezja, senzorji, zasloni LCD in druge naprave.

### 2.2.2 Raspberry Pi Camera Module

Raspberry Pi Camera Module [5] je video kamera, razvita s strani fundacije Raspberry Pi (slika 2.2). Njen namen je dopolnitev računalnika Raspberry Pi z video zmožnostmi. Kamero sestavlja slikovni senzor s petimi milijoni slikovnih točk, objektiv s fiksnim fokusom in integrirano vezje z vrati CSI. Video kamera omogoča računalniku Raspberry Pi zajemanje običajnih fotografij, omogoča pa tudi snemanje video posnetkov z različnimi načini zajetja videa: 1080p @ 30 fps, 720p @ 60 fps in VGA90. Programsko upravljanje video kamere je mogoče z uporabo mnogih Python knjižnic in Bash programov. Ta kamera obstaja tudi v različici, ki ne vsebuje filtra infrardečih žarkov, kar omogoča zajemanje video posnetkov v popolni temi.

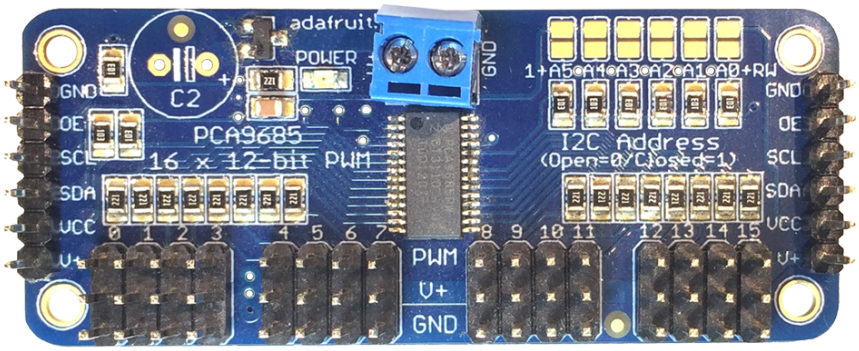



Slika 2.2: Video kamera Raspberry Pi Camera Module.

Video kamera je bila izbrana zaradi zelo majhne velikosti, izjemnih video sposobnosti in relativno nizke cene v primerjavi z ostalimi video kamerami, ki so na voljo na tržišču. Ena od velikih prednosti te kamere je tudi ta, da je posebej namenjena računalniku Raspberry Pi in je zato z njim tudi popolnoma združljiva. Veliko video kamer je le delno združljivih, še več pa popolnoma nezdružljivih. Prednost je tudi način povezave z računalnikom. Mnogo video kamer omogoča povezavo preko priključka USB, ta kamera pa je povezana kar preko vrat CSI. Zaradi majhnega števila priključkov USB na računalniku Raspberry Pi je tak način povezave zelo priročen, saj ne zasede dodatnega priključka USB.

### 2.2.3 Adafruit PWM/Servo Driver

Adafruit 16-Channel 12-bit PWM/Servo Driver [3] je razširitveno vezje za sisteme Arduino in Raspberry Pi. Razvito je v podjetju Adafruit za razširitev možnosti upravljanja naprav PWM preko protokola I2C. Omogoča neodvisno upravljanje do šestnajstih naprav PWM. Posameznemu kanalu lahko nastavimo vrednost signala PWM z dvanajst bitno natančnostjo. Teoretično bi naj bilo možno zaporedno povezati do 62 takih vezij in tako upravljati do




Slika 2.3: Razširitveno vezje Adafruit PWM/Servo Driver.

992 naprav PWM. Razširitveno vezje je vidno na sliki 2.3.

PWM je ena od glavnih tehnik upravljanja servo motorjev in svetlečih diod. Računalnik Rasberry Pi ima na tem področju veliko pomanjkjivost, saj lahko preko nožic GPIO poganja le eno napravo PWM. Zaradi tega je nujno potrebna uporaba razširitvenega vezja (kot je Adafruit PWM/Servo Driver) pri projektih, ki zahtevajo poganjanje večjega števila naprav PWM. Prednost uporabe tega vezja je tudi v tem, da loči napajanje naprav PWM od napajanja računalnika Raspberry Pi. To povzroči potrebo po dveh virih napajanja, vendar zagotovi stabilen vir napajanja računalniku Raspberry Pi ne glede na obremenitev naprav PWM.

Operacijski sistem je dodatni razlog, zaradi katerega je uporaba takega vezja zelo pomembna pri vodenju naprav PWM z uporabo računalnika Raspberry Pi. Visokoravenski operacijski sistemi niso primerni za upravljanje naprav PWM zaradi prekinitev procesne enote in razvrščanja opravil. Upravljanje naprav PWM je časovno zelo intenzivno opravilo, zato potrebuje namensko procesno enoto, ki zagotovi časovno usklajenost signala PWM. Adafruit PWM/Servo Driver je zelo priročno razširitveno vezje, saj z uporabo lastne ure in procesne enote omogoča računalniku Raspberry Pi popolnoma neodvisno upravljanje naprav PWM.

### 2.2.4 Servo motorji in svetleče diode

Za opravljanje zahtevanih opravil potrebuje robot tri servo motorje. Pogonu robota služita dva enaka servo motorja TowerPro SG-5010 [6]. Naklon kamere določa servo motor TowerPro SG-90 [7]. To je manjša in manj zmogljiva različica motorjev uporabljenih za pogon robota. Ta dva tipa motorjev sta bila izbrana zaradi zelo nizke cene, majhne teže in relativno majhne velikosti.

Servo motorja, uporabljena za pogon robota imata omejen kot delovanja na 180°. Za dosego ustreznega delovanja je bilo potrebno servo motor razstaviti, odstraniti notranji potenciometer in ga nadomestiti z dvema 2,2 kΩ uporoma. Potenciometer služi določanju trenutnega kota ročice servo motorja. Ko potenciometer doseže skrajno vrednost, prekine vrtenje ročice. Z nadomestitvijo potenciometra z dvema 2,2 kΩ uporoma je bila dosežena konstantna upornost. S takšno predelavo motor omogoča neprekinjeno vrtenje ročice in opravljanje naloge pogonska motorja. Prav tako kot pri pogonskem motorju je tudi kot delovanja motorja za spreminjanje naklona kamere omejen na 180°. Namen motorja je spreminjanje naklona kamere in s tem vidnega polja robota, zato je kot delovanja 180° povsem zadosten. Predelava tega servo motorja tako ni bila potrebna.

Za osvetlitev vidnega polja video kamere robota so bile uporabljene običajne svetleče diode.

### 2.2.5 Kartica SD

Kartica SD [8] je ključnega pomena za delovanje računalnika Raspberry Pi, saj hrani celoten operacijski sistem in vse uporabniške podatke. Te kartice so v uporabi že mnogo časa, zato jih srečamo v več različnih fizičnih in spominskih velikostih. Računalnik Raspberry Pi uporablja kartice SD standardne velikosti, vendar z uporabo ustreznega adapterja lahko uporabimo tudi novejše kartice manjših velikosti. Računalnik je združljiv z večino kartic SD, uporabljenih v današnjih fotoaparatih in pametnih telefonih pod pogojem, da ima kartica vsaj 2 GB spominske kapacitete.

Kartice se med seboj delijo tudi na različne hitrostne razrede. Razlike med karticami različnih hitrostnih razredov pridejo do izraza šele pri delu z velikimi datotekami. Operacijski sistemi, ki so razviti za računalnik Raspberry Pi, uporabljajo pri svojem delovanju v veliki večini datoteke majhnih velikosti. Razlika v hitrosti je najbolj opazna pri nalaganju operacijskega sistema na kartico in pri posodabljanju programske opreme. Pri običajni uporabi računalnika Raspberry Pi je hitrostni razred uporabljene kartice SD zanemarljiv dejavnik.

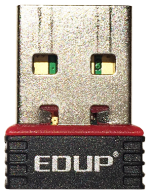
Pri tem projektu sta bili uporabljeni dve kartici SD različnih proizvajalcev in hitrostnih razredov. Ravoj in testiranje robota je potekalo z uporabo kartice Kingston Micro SD 8 GB Class 4 [9]. Končna različica robota pa uporablja hitrejšo kartico Transcend Micro SD 8 GB Class 10 [10]. Glavni razlog za uporabo dveh kartic SD je predvsem ta, da lahko ena kartica opravlja vlogo varnostne kopije. Pri končni različici robota se uporablja hitrejša kartica zaradi malo hitrejšega zagona operacijskega sistema.

### 2.2.6 Sprejemnik WiFi

Žice so pri prenosnih napravah velika ovira. Ena večjih ovir je ethernet kabel, saj naprave omejuje na bližino usmerjevalnika. To je še posebej velik problem pri mobilnih robotih, saj kabli močno omejujejo njihovo prostorsko svobodo.

Raspberry Pi je že tovarniško opremljen z ethernet priključkom, ki omogoča povezavo in komunikacijo na medmrežju. Možnosti za izvedbo komunikacije med računalnikom Raspberry Pi in mobilno aplikacijo je več. Zaradi relativno preproste izvedbe in neodvisnosti po prostorski bližini robota z upravljalcem je bilo za komunikacijo izbrano internetno omrežje. V začetnih fazah razvoja je zadostovala povezava preko ethernet kabla, vendar je to v trenutku, ko je robot postal mobilen, začelo onemogočati nadaljni razvoj in testiranje. Zato je bil kasneje uporabljen sprejemnik WiFi, ki je odpravil potrebo po uporabi ethernet kabla.

Pri izbiri ustreznega sprejemnika je potrebno biti zelo pozoren na zdru-




Slika 2.4: Sprejemnik WiFi - EDUP Ultra-Mini Nano.

žljivost z uporabljenim operacijskim sistemom in strojno opremo računalnika Raspberry Pi. Zaradi majhne velikosti, ugodne cene in združljivosti z uporabljeno opremo je bil izbran sprejemnik EDUP Ultra-Mini Nano Network Adapter [11] prikazan na sliki 2.4.

### 2.2.7 Baterije

Napajalne žice so pri mobilnem robotu prav tako velika ovira, saj mu onemogočajo prosto premikanje po prostoru in zahtevajo prisotnost električne vtičnice. Ta problem odpravimo z uporabo baterijskih virov napajanja.

Skrbna izbira napajalnih baterij je zelo pomemben korak pri načrtovanju robota. Robot mora biti preskrbljen z ustreznim napajanjem, ki mu zagotovi zanesljivo delovanje in tudi ustrezen čas delovanja. Pri preprostih mobilnih robotih z omejeno nosilnostjo je potrebno biti zelo pozoren na težo in velikost uporabljenih baterij.

Zaradi uporabe vezja Adafruit PWM/Servo Driver se je napajanje računalnika in naprav PWM ločilo na dva napajalna vira. Kjub temu, da potrebujemo dve bateriji (kar povzroča dodatno težo in zavzame več prostora) imata dva vira napajanja veliko prednost. Za napajanje računalnika Raspberry Pi in vseh perifernih enot se tako uporablja en vir napajanja, za servo motorje pa drugi vir napajanja. Servo motorji pri povprečni uporabi potrebujejo okrog 150 mA električnega toka. Za tri servo motorje potrebujemo približno 450 mA električnega toka. Ob visoki obremenitvi servo motorjev se njihova poraba električnega toka še dodatno poviša, lahko tudi do enega ali celo dveh amperov. Takšno nihanje v električnem toku lahko

vodi do situacije, ko napajalni vir ne zmore napajati obeh potrošnikov, kar lahko povzroči zaustavitev sistema računalnika Rasbperry Pi.

Dodatna slabost uporabe enega vira napajanja je ta, da poteka napajanje servo motorjev preko nožic GPIO računalnika Raspberry Pi. Pri veliki obremenitvi motorjev bi lahko računalnik celo pregorel. Zaradi teh razlogov sta dva vira napajanja zaželena.

Servo motorji in svetleče diode potrebujejo za optimalno delovanje od 4,5 V do 6 V električne napetosti. Za njihovo napajanje so se odlično izkazale alkalne baterije napetosti 1,5 V. Štiri zaporedno vezane baterije so tako skupaj proizvedle 6 V električne napetosti. Za alkalne baterije je značilno, da se jim pri uporabi postopoma manjša električna napetost. Njihovo območje delovanja je običajno omejeno med 1,5 V (popolnoma napolnjene) in 1,2 V (izpraznjene). Tik pred izpraznitvijo baterije še vedno ustvarjajo približno 4,8 V (4 * 1,2 V) električne napetosti, kar je ravno nad zahtevano mejo servo motorjev. Zaradi teh značilnosti se je uporaba štirih alkalnih baterij izkazala za odlično rešitev, saj ne glede na izpraznjenost proizvajajo električno napetost, ki zadošča napajanju servo motorjev in svetlečih diod.

Nekoliko bolj problematično je napajanje računalnika Raspberry Pi, saj za razliko od servo motorjev in svetlečih diod potrebuje stalno in zanesljivo napajanje napetosti 5 V (±5%). Previsoka električna napetost lahko škoduje vezju, prenizka pa ne dovaja ustrezne napetosti za delovanje. LiPo baterije (angl. *lithium polymer battery*), ki so običajno uporabljene pri raznih hobi projektih, ne proizvedejo ustrezne električne napetosti. Te baterije so v večini primerov zelo drage, potrebujejo pa tudi posebne polnilne postaje [12]. Zaradi teh razlogov so neprimerne za uporabo pri projektih z nizkim proračunom.

Kot rešitev se ponovno ponudijo alkalne baterije. Zaradi manjšanja napetosti alkalnih baterij ob izpraznitvi, je potrebno uporabiti večje število baterij v kombinaciji z regulatorjem UBEC [13]. UBEC je regulator električne napetosti, ki omeji napetost baterije na porabnikovi strani. Tako lahko napetost petih alkalnih baterij, ki proizvedejo od 7,5 V (5 * 1,5 V) do 6 V (5 * 1,2 V)

električne napetosti, reguliramo na konstantnih 5 V napetosti. Regulatorji UBEC so zelo učinkoviti, saj je izguba električne energije ob njihovi uporabi minimalna.

Drugo rešitev predstavlja uporaba prenosne polnilne postaje za mobilne telefone [12]. Večina takih polnilnih postaj ima izhodno napetost ravno 5 V. V primerjavi z alkalnimi baterijam in regulatorji UBEC imajo nekaj prednosti: zavzamejo manj prostora, imajo manjšo skupno težo in so cenovno bolj ugodne. Poleg tega ponujajo take postaje tudi možnost polnjenja, kar je pri baterijskih virih napajanja vedno dobrodošla značilnost. Zaradi vseh prednosti je bila za napajanje računalnika Raspberry Pi uporabljena prenosna polnilna postaja kapacitete 2200 mAh, izhodnega toka 1 A in izhodne napetostje 5 V.

Kombinacija uporabe običajnih alkalnih baterij in polnilne postaje za mobilne telefone se je izkazala za zelo poceni in zanesljivo rešitev.

## 2.3 Programska oprema

Poleg strojne opreme ima tudi programska oprema pomembno vlogo pri razvoju robota. V mnogih primerih je izbira programske opreme neposredno pogojena uporabljeni strojni opremi. Tudi v tem primeru je bila večina programske opreme izbrane ravno zaradi združljivosti s strojno opremo oziroma zaradi stalnih praks in dogovorov, ki veljajo na določeni pratformi.

Fundacija Raspberry Pi je kot glavni razvojni jezik računalnika določila programski jezik Python. Posledično uporablja velika večina razširitvenih vezij in dodatkov namenjenih računalniku Raspberry Pi ravno knjižnice napisane v jeziku Python, zaradi tega je tudi programska oprema za vodenje robota in komunikacijo z mobilno aplikacijo razvita v Pythonu. Zaradi združljivosti z uporabljeno knjižnico za vodenje naprav PWM je bila uporabljena različica Python v2.7. Pri konfiguraciji programske opreme se je v veliki meri uporabljala tudi lupina Bash. Za razvoj mobilne aplikacije je bila izbrana platforma iOS [14]. Ta odločitev je pogojevala uporabo programskega jezika Objective-C, paketa za razvoj programske opreme iOS SDK in

razvojnega okolja Xcode [15].

### 2.3.1 Occidentalis

Med množico operacijskih sistemov za računalnik Raspberry Pi je bil pri tem projektu izbran operacijski sistem Occidentalis v0.2 [16]. Razvilo ga je podjetje Adafruit za doseganje višje združljivosti in lažje uporabe računalnika Raspberry Pi ob uporabi njihovih razširitvenih vezij. Mnoga razširitvena vezja tega podjetja potrebujejo za pravilno delovanje nastavitev dodatne programske opreme. Z uporabo tega operacijskega sistema lahko večino teh vezij uprabljamo brez predhodne nastavitve, kar olajša začetne korake razvoja nekega projekta. Priloženih ima tudi mnogo orodij, ki olajšajo uporabo računalnika Raspberry Pi. Occidentalis je nadgradnja operacijskega sistema Raspbian Wheezy [17], ki temelji na Debian Wheezy [18] distribuciji Linux operacijskega sistema prilagojenega za arhitekturo ARM. Operacijski sistem Occidentalis je bil izbran ravno zaradi velikega števila orodij, ki jih po privzetem vsebuje in zaradi uporabe razširitvenega vezja Adafruit PWM/Servo Driver.

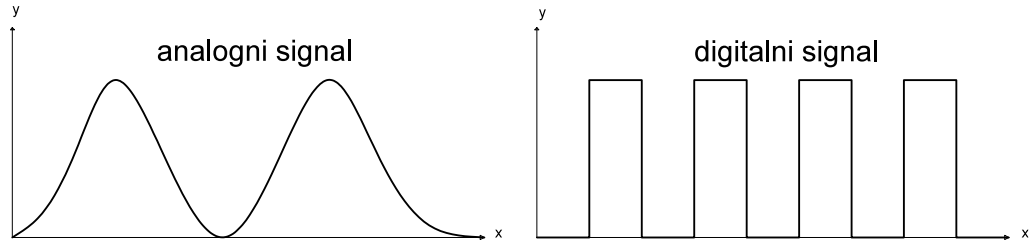
### 2.3.2 GStreamer

GStreamer [19] je večpredstavno programsko ogrodje, ki omogoča delo z avdio in video datotekami ter prenašanje pretočnega videa preko internetnega omrežja. Prednost programa GStreamer je doseganje prenosa pretočnega videa z zelo nizko zakasnitvijo. Ta je ključnega pomena pri vodenju robota, saj lahko upravljalec le tako uspešno vodi robota. Za doseganje tako nizke zakasnitve prenosa pretočnega videa je potrebna uporaba programskega ogrodja GStreamer tako na strežniku kot na odjemalcu. Na računalniku Raspberry Pi je bilo uporabljeno programsko ogrodje GStreamer različice v1.0, s katerim je bil vzpostavljen strežnik pretočnega videa. Na odjemalčevi strani (mobilna aplikacija) pa je bilo uporabljeno ogrodje GStreamer iOS SDK, ki omogoča vzpostavitev povezave s strežnikom in predvajanje pretočnega videa.

## 2.4 Tehnologije

### 2.4.1 Pulzno-širinska modulacija

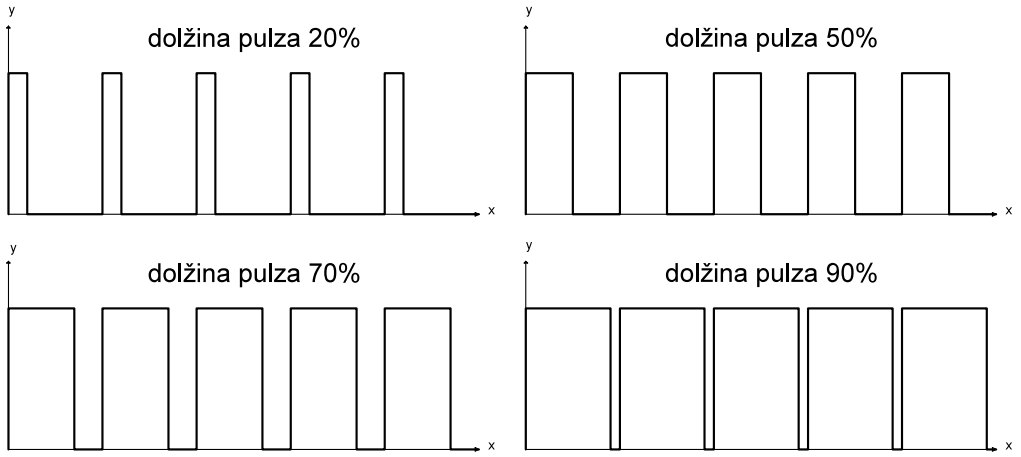
Pulzno-širinska modulacija (angl. *pulse-width modulation* - PWM) je tehnika, ki nam omogoča predstavitev digitalnega signala v analogni obliki [20]. Digitalni signal je diskretni signal kvadratne oblike, ki lahko zajema vrednosti 0 ali 1. V primeru napajanja servo motorja z električno napetostjo 5 V, predstavljata ti dve vrednosti 0 ali 5 V. To pomeni, da bo ročica servo motorja pri vrednosti signala 0 (signal v nizkem stanju, napetost 0 V) mirovala, pri vrednosti signala 1 (signal v visokem stanju, napetost 5 V) pa se bo obračala s polno hitrostjo. Takšno upravljanje servo motorjev ni najbolj ugodno, saj bi servo motorju želeli določati različno hitrost obračanja ročice. To lahko dosežemo z uporabo analognega signala.




Slika 2.5: Primer analognega in digitalnega signala.

Analogni signal je zvezni signal sinusne oblike, ki lahko na nekem določenem intervalu zajema neomejeno število vrednosti. Z uporabo analognega signala lahko servo motorju dovajamo poljubno električno napetost med 0 in 5 V (v primeru uporabe napajalnega vira električne napetosti 5 V), kar mu omogoča vrtenje ročice z različno hitrostjo. Uporaba ustreznega potenciometra omogoča reguliranje hitrosti servo motorja tudi pri digitalnem signalu. Z obračanjem ročice potenciometra spreminjamo njegovo električno napetost, kar omogoča servo motorju doseganje različnih hitrosti. Primera analognega in digitalnega signala sta vidna na sliki 2.5.

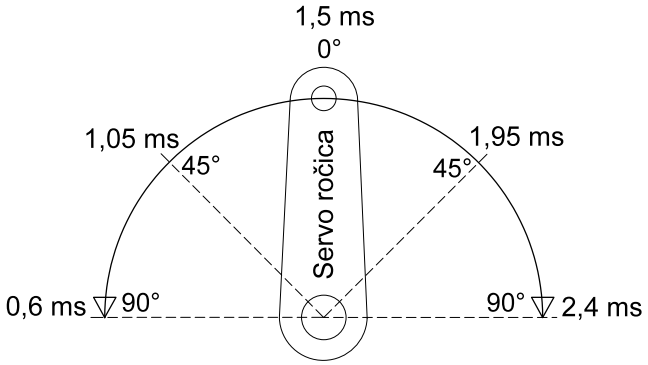
Pri uporabi digitalnih vezij smo vedno omejeni na digitalni signal, zato




Slika 2.6: Primeri signala PWM pri različni dolžini pulza.

potrebujemo način pretvarjanja digitalnega signala v analogni signal oziroma predstavitev digitalnega signala v analogni obliki (v primeru uporabe PWM). Modulacija PWM je definirana z dvema parametroma: frekvenco preklapljanja in dolžino pulza. Frekvenca preklapljanja določa število pulzov oziroma period v eni minuti, dolžina pulza pa predstavlja delež časa ene periode, ko je pulz v visokem stanju. Signal pulza lahko v eni periodi zajema visoko in nizko stanje. V primeru uporabe napajalnega vira s 5 V električne napetosti in modulacije PWM z 10% dolžino pulza (pulz je v visokem stanju 10% časa periode), dosežemo napajanje porabnika z 0,5 V, pri 50% dolžini pulza pa 2,5 V. Dovajanje napetosti 5 V v zelo hitrih intervalih pri različni dolžini pulza nam omogoči upravljanje hitrosti vrtenja ročice servo motorja. Slika 2.6 prikazuje signale PWM z različnimi dolžinami pulzov.

Uporaba modulacije PWM se nekoliko razlikuje pri upravljanju servo motorjev ali svetlečih diod. Servo motorji v večini uporabljajo standardiziran način delovanja. Delovanje ročice servo motorja je običajno omejeno na kot 180°. Za nastavitev določenega kota ročice je potrebno uporabiti signal PWM z ustreznimi parametri. Večina servo motorjev potrebuje za ustrezno delovanje frekvenco 20 ms (milisekund), dolžino pulza pa med 0,6 in 2,4 ms. Spre-



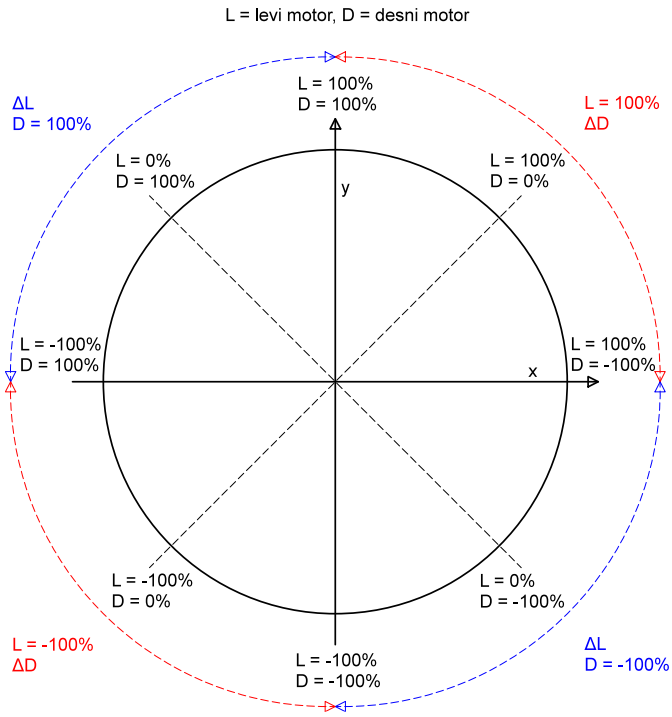

Slika 2.7: Stanje ročice motorja ob različnih dolžinah pulza signala PWM.

minjanje dolžine pulza med tema vrednostima omogoča spreminjanje kota ročice servo motorja. Signal z dolžino pulza 1,5 ms predstavlja nevtralno pozicijo servo motorja, ročica je nastavljenja na kot 0°. Slika 2.7 prikazuje stanje ročice servo motorja ob različnih dolžinah pulza signala PWM. Servo motorji, ki omogočajno neomejeno obračanje ročice, delujejo na enak način. Nastavljanje dolžine pulza med 0,6 in 2,4 ms določa smer in hitrost vrtenja ročice. Vrednosti dolžine pulza pod 1,5 ms povzročijo vrtenje ročice v eno smer, vrednosti nad to mejo pa povzročajo vrtenje ročice v nasporotno smer. Z bližanjem nastavljene dolžine pulza zgornji ali spodnji meji se hitrost vrtenja ročice povečuje. Ob nastavitvi dolžine pulza na 1,5 ms se ročica motorja preneha obračati.

Upravljanje svetlečih diod je bolj preprosto. Dolžina pulza lahko zajema vrednosti na celotnem intervalu med 0 in 20 ms, če je frekvenca enaka 20 ms. Z reguliranjem dolžine pulza določamo svetilnost svetlečih diod.

### 2.4.2 Diferencialni krmilni sistem

Diferencialni krmilni sistem (angl. *Differential Steering System*) je zelo priljuben način krmiljenja robotov [21]. Za izvedbo takega načina krmiljenja potrebujemo le dve pogonski kolesi. V mnogih primerih se ob uporabi diferencialnega krmilnega sistema uporablja tudi eno stabilizacijsko vsesmerno




Slika 2.8: Diferenčni krmilni sistem, ponazorjen na koordinatnem sistemu.

kolo. Prednosti takšnega krmiljenja so relativno preprosta izvedba, uporaba manjšega števila servo motorjev in posledično tudi manjša teža in nižja poraba električne energije.

Robota bi bilo mogoče krmiliti z uporabo ločenega krmilnega sistema (podobno kot je krmiljenje izvedeno v avtomobilu), ki je sicer izvedljivo, vendar je za preproste robote neprimerno, saj je mnogo bolj zahtevno in hkrati zahteva uporabo večjega števila servo motorjev in koles ter uporabo dodatne mehanike za izvedbo dejanskega krmilnega sistema.

Z uporabo dveh pogonskih koles, ki se obračata z različno hitrostjo, lahko sočasno upravljamo poganjanje in krmiljenje robota. Robot vozi v ravni liniji, ko se obe kolesi vrtita z enako hitrostjo v enaki smeri. Za doseganje desnega zavoja je potrebno zmanjšati hitrost desnega motorja, ali povečati hitrost levega motorja. Hitrost posameznega kolesa določa, kako oster zavoj

bo robot izpeljal. S spreminjanjem hitrosti koles premikamo točko na osi koles, okrog katere se robot vrti.

Delovanje krmiljenja, razvitega na robotu lahko prikažemo z uporabo koordinatnega sistema in enotske krožnice. Hitrost posameznega motorja je določena glede na kvadrant koordinatnega sistema, v katerem se nahaja kazalec igralnega ploščka uporabniškega vmesnika (mobilna aplikacija). V posameznem kvadrantu ima en motor konstantno hitrost, drugemu motorju pa se hitrost spreminja, kar določa ostrino zavoja. Prvi kvadrant koordinatnega sistema predstavlja desno zavijanje robota. V tem kvadrantu se spreminja hitrost desnega motorja, hitrost levega motorja pa je konstantna (100%). Hitrost desnega motorja se spreminja od 100% (na enotski krožnici prikazano s kotom $\pi/2$), kjer robot vozi v ravni liniji, proti 0% (kot $\pi/4$), kjer se robot vrti okrog osi desnega kolesa, do -100% (kot 0), kjer se robot vrti okrog svoje osi. Delovanje diferencialnega krmilnega sistema je predstavljeno na sliki 2.8.

## 2.5 Razvoj

Ko je bila faza načrtovanja zakjučena, se je začela izvedba začrtane ideje. Razvoj robota je bil sistematičen in je postopoma napredoval od osnovnih proti vse bolj naprednim funkcijam. Razvoj programske opreme je potekal v štirih stopnjah. Prva stopnja je vkjučevala pisanje pomožnih Python razredov, ki omogočajo upravljanje naprav PWM in njihovo testiranje. Druga stopnja je predstavljala razvoj mobilne aplikacije za vodenje robota. V tretji stopnji je bila izvedena komunikacija med robotom in mobilno aplikacijo ter glavna logika za vodenje robota. Četrta stopnja je zajemala izvedbo video prenosa. Razvoj robota se je začel z osnovno vzpostavitvijo računalnika Raspberry Pi.

### 2.5.1 Priprava

#### 2.5.1.1 Priprava kartice SD

Prvi korak razvoja je bila priprava kartice SD, saj jo računalnik Raspberry Pi nujno potrebuje za svoje delovanje. Operacijski sistem lahko naložimo na

kartico SD z uporabo lupinskega programa `dd`. Njegova uporaba je zelo preprosta: kot vhodne podatke mu podamo pot do operacijskega sistema in lokacijo, kamor jo želimo naložiti (v tem primeru pot do kartice SD) [22].

Ob prvem zagonu moramo računalnik Raspberry Pi priključiti s tipkovnico in ethernet kablom ter povezati na zunanji zaslon, saj operacijski sistem zahteva dodatno konfiguracijo. Pred uporabo sistema se je potrebno prijaviti z uporabniškim imenom `pi` in geslom `raspberry`. Do zaslona z osnovno konfiguracijo lahko dostopamo z lupinskim ukazom `raspi-config`. Dve pomembnejši nastavitvi sta razširitev uporabljenega prostora na kartici SD in vkjučitev omrežnega protokola SSH.

Razširitev uporabljenega prostora na kartici SD je pomembna predvsem pri uporabi kartic SD s 4 GB pomnilniškega prostora ali več. Operacijski sistem privzeto zazna samo 2 GB pomnilniškega prostora. Z opcijo `Expand Filesystem` na konfiguracijskem zaslonu dosežemo uporabo celotnega pomnilniškega prostora kartice SD. Z izbiro opcije `Advanced Options` se pomaknemo v meni z naprednimi nastavitvami, kjer omogočimo protokol SSH z izbiro možnosti `SSH`. Ta močno olajša uporabo računalnika Raspberry Pi, saj omogoča prijavo v lupino računalnika kar preko omrežja. Tako se izognemo potrebi po uporabi tipkovnice in zunanjega zaslona, saj lahko računalnik Raspberry Pi uporabljamo kar preko prenosnega računalnika.

Ker operacijski sistem uporablja privzeto prijavno geslo `raspberry`, ga je iz varnostnih razlogov priporočljivo spremeniti. To storimo z izbiro možnosti `Change User Password`, ki zahteva vnos novega gesla. Računalnik nato ponovno zaženemo z ukazom `reboot`.

Dodaten priporočljiv korak pri vsakem sveže nameščenem operacijskem sistemu je tudi posodobitev programske opreme. Posodobitev sprožimo z ukazom `apt-get update` in `apt-get upgrade`. Prvi ukaz sproži posodobitev seznama uporabljenih paketov, drugi pa izvede dejansko posodobitev. Nalaganje operacijskega sistema na kartico SD in posodabljanje programske opreme sta edina koraka, pri katerih je moč opaziti razliko med hitrostnimi razredi uporabljenih kartic SD. Pri uporabi kartice SD hitrostnega razreda

4 je posodobitev trajala približno 4 ure, pri uporabi kartice SD hitrostnega razreda 10 pa le slabo uro. Po opravljeni konfiguraciji in posodobitvi programske opreme se lahko začne dejanski razvoj.

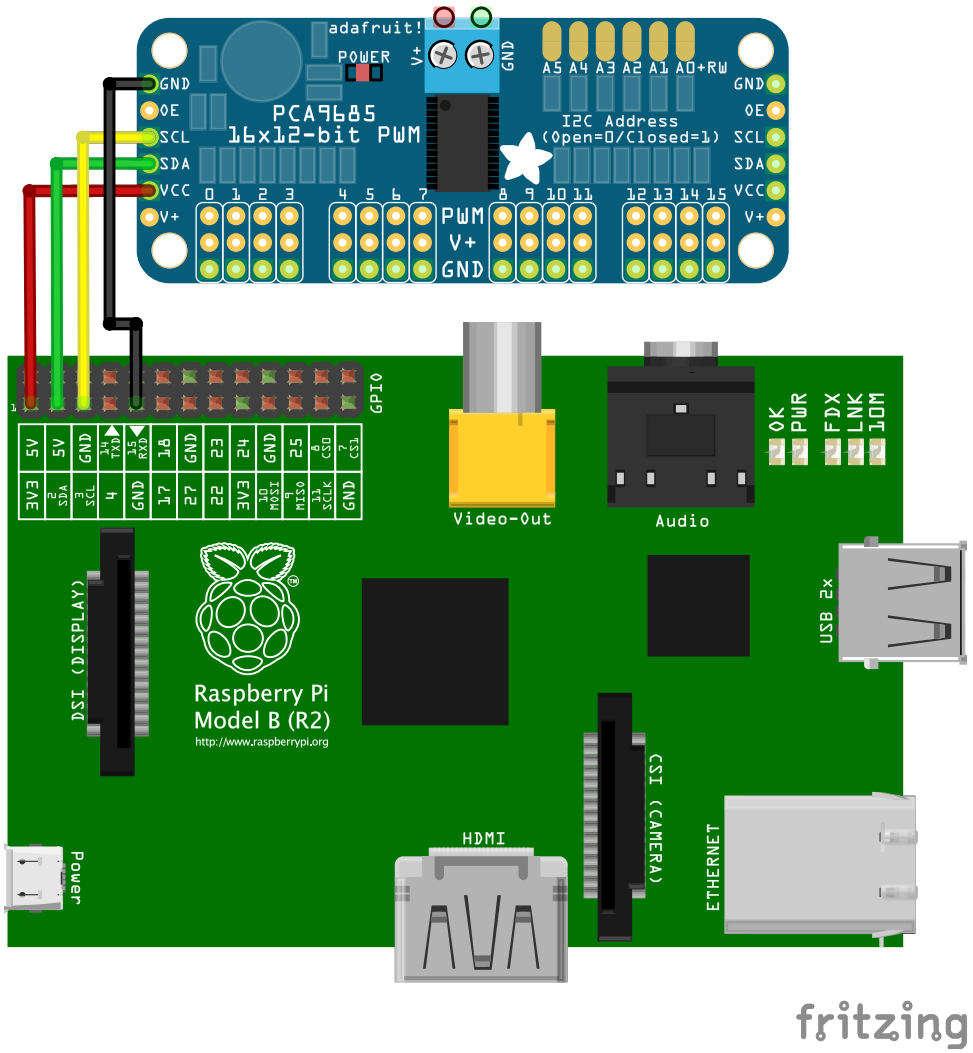
#### 2.5.1.2 Povezava naprav PWM

Razvoj robota se je začel s povezavo in testiranjem delovanja servo motorjev in svetlečih diod. Za njihovo upravljanje je bilo uporabljeno razširitveno vezje Adafruit PWM/Servo Driver. Sporazumevanje računalnika z vezjem zahteva dodatno programsko opremo (`python-smbus` in `i2c-tools`). Operacijski sistem Occidentalis že privzeto vsebuje potrebovano programsko opremo, zato namestitev ni bila potrebna. Pred uporabo je potrebno razširitveno vezje pravilno povezati z nožicami GPIO računalnika Raspberry Pi. Nožico `VCC` razširitvenega vezja je potrebno povezati z nožico GPIO št. `1`, nožico `SDA` z nožico št. `3`, nožico `SCL` z nožico št. `5` in nožico `GND` z nožico št. `9`. Povezava razširitvenega vezja z nožicami GPIO je prikazana na sliki 2.9.

Ko je razširitveno vezje ustrezno povezano z nožicami GPIO, lahko na poljubni kanal vezja povežemo servo motor in preizkusimo delovanje [23]. Za upravljanje naprav PWM potrebujemo še Python knjižnico [24] za delo z razširitvenim vezjem. Knjižnica vsebuje tudi testne programe, s katerimi lahko preizkusimo delovanje povezav.

### 2.5.2 Programiranje

Za olajšanje postopka razvoja programske opreme za vodenje robota sta bili uporabljeni razvojni okolji PyCharm [25] in Xcode [15]. Ker nista podprti s strani operacijskega sistema Occidentalis, poleg tega pa bi bil tudi razvoj programske kode na dejanskem računalniku Raspberry Pi zelo zamuden in dolgotrajen, je bila programska koda napisana na prenosnem računalniku. Naveden način razvoja je pred vsakim testiranjem zahteval najprej prenašanje kode na računalnik Raspberry Pi. V izogib temu zamudnemu procesu je bil uporabljen protokol SFTP [26]. Na računalnik Raspberry Pi je bil nameščen odjemalec SSHFS, ki se je povezal s strežnikom SSHFS na prenosnem računalniku [27]. Opisana povezava je omogočala po-




Slika 2.9: Povezava nožic GPIO z razširitvenim vezjem.

ganjanje programske kode kar na računalniku Raspberry Pi kjub temu, da se je koda dejansko nahajala na prenosnem računalniku. Izvajanje programa na računalniku Raspberry Pi je tako potekalo iz prenosnega računalnika preko povezave SSH. Opisan način dela je prenosnemu računalniku omogočil popolni nadzor nad računalnikom Raspberry Pi in je proces razvoja programske opreme precej olajšal in pospešil.

#### 2.5.2.1 Upravljanje naprav PWM

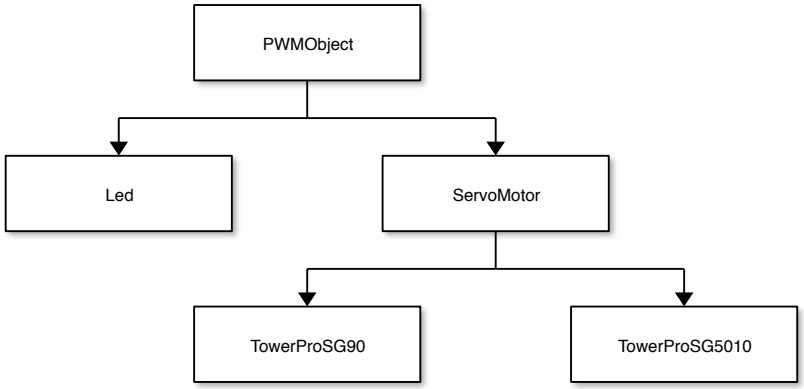
Prvi korak pri pisanju programa za vodenje robota je bil razvoj pomožnih Python razredov za delo z napravami PWM. Razredi se hierarhično delijo od splošnih proti strogo namenskim razredom, ki služijo upravljanju določene naprave PWM. Hierarhija razredov je predstavljena na sliki 2.10.

Ti razredi močno poenostavijo inicializacijo in vodenje naprav PWM. Glavni razred v hierarhiji je `PWMObject`. Ta predstavlja splošno napravo PWM in vsebuje metode za določanje časovnega intervala, v katerem je signal PWM v visokem stanju. Ob inicializaciji mu kot parametre podamo objekt PWM in številko kanala razširitvenega vezja, na katerega je naprava povezana. Objekt PWM izhaja iz podporne knjižnice razširitvenega vezja in je skupen za vse uporabljene naprave PWM.

Iz razreda `PWMObject` sta izpeljana dva podrazreda: `Led` in `ServoMotor`. Razred `Led` predstavlja svetlečo diodo. Njeno upravljanje poenostavi do te mere, da lahko svetilnost nadzorujemo kar preko klica metode `set_brightness()`, ki prejme kot argument svetilnost predstavljeno v odstotkih. Upravljanje svetleče diode je tako popolnoma prilagojeno funkciji, ki jo ta opravlja. Razred `ServoMotor` predstavlja splošni servo motor. Poleg objekta PWM in kanala mu ob inicializaciji podamo tudi podatke o tem, ali se lahko ročica motorja vrti neprekinjeno in ali naj se vrti v obratni smeri.

Razred `TowerProSG90` deduje od razreda `ServoMotor` in predstavlja dejanski servo motor TowerPro SG 90. Njegov namen je nastavljanje naklona kamere, zato vsebuje metodo `set_angle()`. Ta metoda prejme kot parameter številko, ki predstavlja kot, na katerega želimo nastaviti ročico servo motorja.

Za pogon robota skrbita motorja TowerPro SG 5010, ki ju upravljamo




Slika 2.10: Hierarhija pomožnih Python razredov.

preko razreda `TowerProSG5010`. Poleg objekta PWM in številke kanala prejme ta objekt ob inicializaciji tudi podatke o umeritvi motorja in njegovi legi. Možne postavitve motorja so določene v razredu s konstantami. Vrednosti so `MotorPosition.NONE = 0`, `MotorPosition.RIGHT = 1` in `MotorPosition.LEFT = 2`. Glede na to, ali se motor nahaja na levi ali na desni strani, prejme pri inicializaciji ustrezno konstanto. Ta podatek določa smer, v katero se bo vrtela ročica motorja (pogonska motorja sta drug drugemu zrcalno nameščena, zato se ročici vrtita v obratni smeri). Uporabljena motorja sta nižjega cenovnega razreda, zato se ob enakih vrednostih signala PWM ročici vrtita z različno hitrostjo. To povzroča potrebo po uporabi umeritvenih podatkov. Ti so predstavljeni v obliki konstant, hranjenih v ločenih razredih. Vsak od dveh motorjev ima svoj razred s podatki, ki ga razredu `TowerProSG5010` ob inicializaciji podamo kot parameter.

Za testiranje delovanja razvitih razredov so bili ustvarjeni trije testni razredi, ki so v različnih časovnih intervalih nastavljali svetilnost svetlečih diod, kot ročice motorja za nastavitev naklona kamere in hitrost ter smer vrtenja pogonskih motorjev. Posamezni razred je testiral eno napravo. Ker uporabljene naprave nimajo senzorjev za spremljanje trenutnega stanja naprave

(ni povratne informacije), je potrebno tak način testiranja ročno nadzorovati. Ob vsaki spremembi je bilo potrebno preveriti, ali stanje naprave ustreza nastavljenemu stanju v programu.

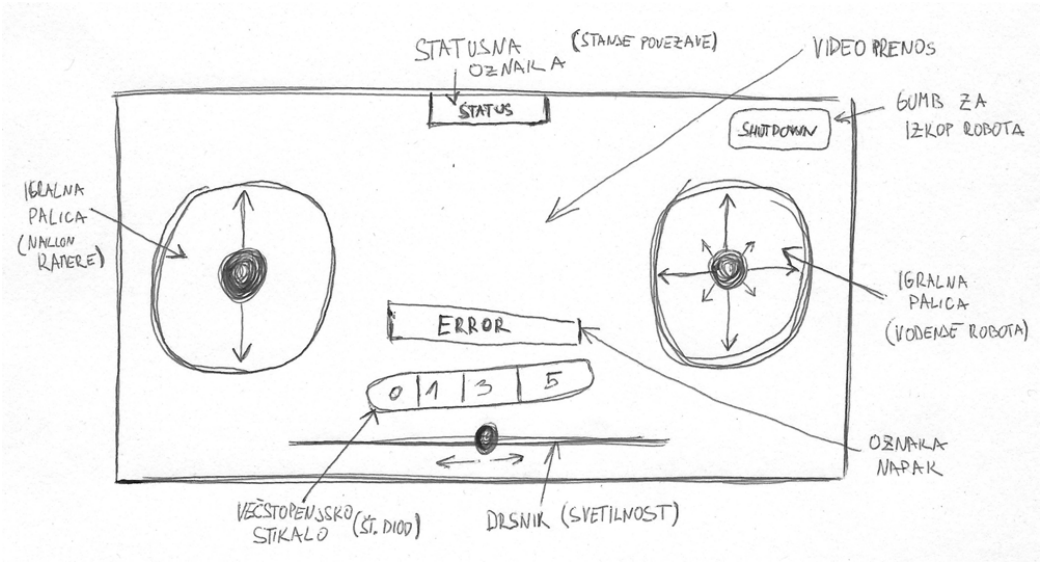
Sledila je izvedba razreda `Robot`. Namen tega razreda je upravljanje vseh naprav PWM in sporazumevanje z mobilno aplikacijo. Na tej stopnji mobilna aplikacija še ni bila razvita, zato način sporazumevanja ni bil točno definiran. Iz navedenega razloga je razred `Robot` zelo preprost. V neskončni `while` zanki bere ukaze s standardnega vhoda in ustrezno upravlja delovanje naprav. Primer: ukaz `1 100` sproži vrtenje levega pogonskega motorja s polno hitrostjo, ukaz `2 50` pa sproži vrtenje desnega pogonskega motorja s polovično hitrostjo.

#### 2.5.2.2 Mobilna aplikacija

Sledil je razvoj mobilne aplikacije za platformo iOS. Razvita je bila z uporabo razvojnega paketa iOS SDK in razvojnega okolja Xcode. Aplikacija je univerzalna, kar pomeni, da je njen uporabniški vmesnik prilagojen vsem napravam, ki poganjajo operacijski sistem iOS.

Prvi korak je zajemal načrtovanje uporabniškega vmesnika aplikacije in načina uporabe elementov. Slika 2.11 prikazuje papirnati prototip uporabniškega vmesnika. Glavna naloga aplikacije je vodenje robota po prostoru, nastavljanje naklona video kamere in upravljanje svetilnosti svetlečih diod. Poleg svetilnosti želimo nadzirati tudi število prižganih svetlečih diod. Z aplikacijo nadzorujemo tri različne funkcije robota, pri čemer se upravljanje svetlečih diod deli na dve ločeni opravili. To pomeni, da uporabniški vmesnik potrebuje štiri elemente, preko katerih bo takšna vrsta upravljanja mogoča.

Glede na funkcijo zahtevajo elementi različne načine uporabe. Pri upravljanju svetilnosti svetlečih diod in naklona kamere nas zanima samo linearna sprememba in sicer svetilnost na intervalu od 0 do 100 oziroma kot na intervalu med -90 in 90. Tak način uporabe najbolje opisuje element drsnika. Možnosti za izbiro števila prižganih svetlečih diod se delijo na 0, 1, 3 ali 5, zato lahko tak način uporabe dosežemo z uporabo štiristopenjskega sti-




Slika 2.11: Papirnati prototip uporabniškega vmesnika.

kala. Vsaka stopnja določa ustrezno število prižganih svetlečih diod. Vodenje robota po prostoru zahteva dva podatka: hitrost in smer. Te podatke se da prikazati s koordinatnim sistemom, na katerem oddaljenost od izhodišča predstavlja hitrost, kot z osjo `Y` pa smer vožnje. Upravljanje je podobno uporabi igralne palice igralnih konzol. Ker tak način uporabe ni najbolj običajen, je bilo potrebno element igralne palice razviti lastnoročno. Rezultat je bil uporabniški vmesnik z dvema drsnikoma, enim štiristopenjskim stikalom in eno igralno palico.

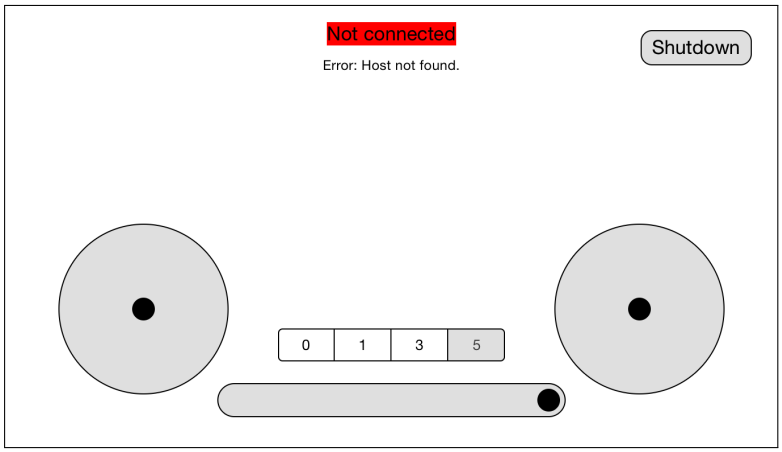
Načrtovanju je sledila izvedba uporabniškega vmesnika in razvoj elementa igralne palice. Razlika med delovanjem drsnika in igralne palice je le v premikanju kazalca po eni ali dveh oseh koordinatnega sistema, zato je bila za izvedbo obeh elementov uporabljena po meri razvita komponenta `ControlView`. Ta odločitev je bila sprejeta z namenom doseganja enotnega izgleda in delovanja elementov.

Komponento `ControlView` sestavljata črni kazalec, ki predstavlja izbrano vrednost in v ozadju izrisan krog, ki definira območje premika kazalca. Delovanje komponente je odvisno od tipa, ki ji ga nastavimo ob inicializaciji.

Ob nastavitvi tipa igralne palice omogoča komponenta prosto premikanje kazalca po koordinatnem sistemu. Oddaljenost kazalca od sredine predstavlja hitrost robota, njegov kot z osjo `Y` pa smer potovanja. Ob izpustu kazalca se ta samodejno vrne na koordinati `(0, 0)`, kar ustreza ustavitvi robota. Ponastavitev vrednosti je zaželjena samo pri igralni palici, zato drsniki ob izpustu kazalca ohranijo vrednosti. Razlika med drsnikoma za nastavljanje svetilnosti svetlečih diod in naklona kamere je le v začetni postavitvi kazalca. Oba kazalca sta sprva v nevtralni postavitvi, vendar se ta med drsnikoma razlikuje. Nevtralna postavitev kazalca drsnika za nastavljanje naklona kamere je na sredini drsnika (vrednost 0 na itervalu -90, 90). Pri drsniku za nastavljanje svetilnosti svetlečih diod pa je nevtralna pozicija kazalca na levem robu drsnika (vrednost 0 na itervalu 0 do 100). Drsnik za nastavljanje svetilnosti svetlečih diod je običajne podolgovate oblike. Iz estetskih razlogov je drsnik za nastavitev kota kamere enake oblike in velikosti kot element igralne palice.

Za nastavljanje števila prižganih svetlečih diod je bilo uporabljeno običajno večstopenjsko stikalo tipa `UISegmentedControl`, njegov izgled je bil prilagojen izgledu komponente `ControlView`. Nastavljanje svetilnosti svetlečih diod je onemogočeno v primeru, ko je izbrana vrednost stikala `0`, zato je v tem stanju drsnik skrit. Ob izbiri možnosti, kjer je prižgana vsaj ena svetleča dioda, se drsnik prikaže in tako omogoči nastavljanje svetilnosti.

Igralna palica in drsnik za nastavljanje kota kamere sta postavljena ob desnem in levem robu zaslona. Njuna uporaba je namenjena desnemu in levemu palcu. Stikalo in drsnik za upravljanje svetlečih diod sta postavljena na spodnji rob zaslona. Tako ostaja srednji del zaslona neprekrit. Elementi so obrobljeni s črno barvo, njihovo ozadje pa je obarvano s polprosojno sivo barvo. Razlog za opisano barvno kombinacijo in postavitev elementov je boljša vidljivost ozadja aplikacije, saj bo ta prikazoval video prenos robota. Celoten uporabniški vmesnik je zasnovan z namenom oponašanja razporeditve elementov igralnega ploščka. Realiziran uporabniški vmesnik je viden na sliki 2.12.



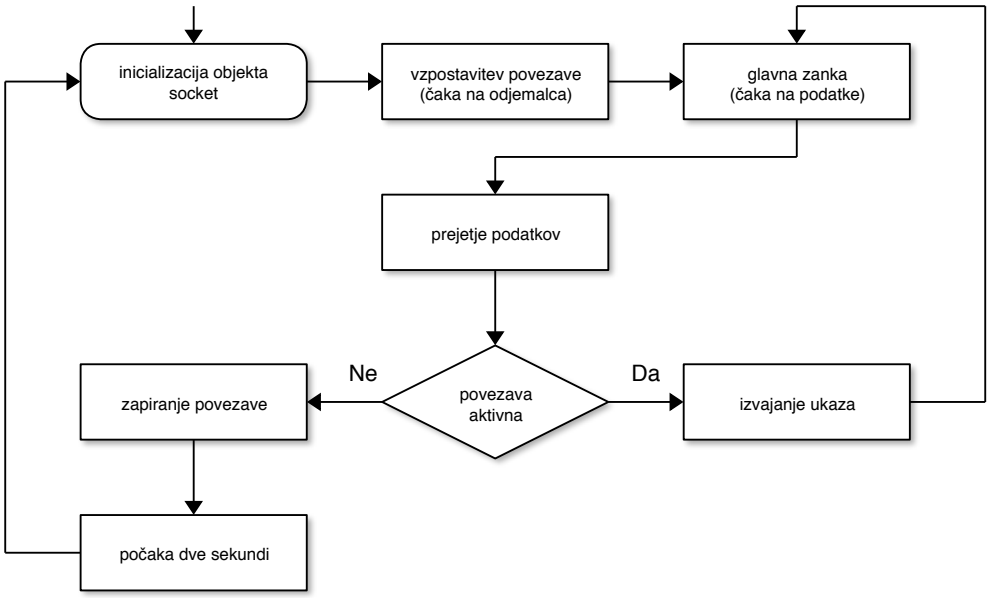

Slika 2.12: Realiziran uporabniški vmesnik.

Uporabniški vmesnik je bil v kasnejših stopnjah razvoja nadgrajen z dodatnimi elementi. Nadgradnje uporabniškega vmesnika so se izvajale sočasno z razvojem in izvedbo funkcionalnosti sporazumevanja z robotom in prikaza njegovega video prenosa.

#### 2.5.2.3 Sporazumevanje odjemalca s strežnikom

Na tej stopnji sta tako robot kot mobilna aplikcija razvita do te mere, da je mogoča izvedba medsebojnega sporazumevanja. Ta temelji na modelu odjemalec/strežnik. Robot opravlja vlogo strežnika, na katerega se v vlogi odjemalca poveže mobilna aplikacija. Vzpostavitev medsebojnega sporazumevanja močno olajša uporaba protokola Bonjour/Zeroconf [28]. Ta protokol omogoča preslikavo omrežnega imena naprave v njen naslov IP. Tako se lahko na robota povežemo tudi v primeru, če naslova IP ne poznamo. Protokol Bonjour/Zeroconf omogoča vzpostavitev povezave kar z uporabo omrežnega imena naprave ter pripone `.local`. V primeru povezave na robota lahko uporabimo naslov `rpi.local` (rpi je omrežno ime robota). Potek komunikacije na strežniku je opisan v naslednjem odstavku.

Razred `Robot` v konstruktorju inicializira objekt vtič (angl. *socket*) [29]. Vtič je programska osnova medprocesne komunikacije. Komunikacija lahko poteka med dvema lokalnima procesoma ali med dvema oddaljenima proce-



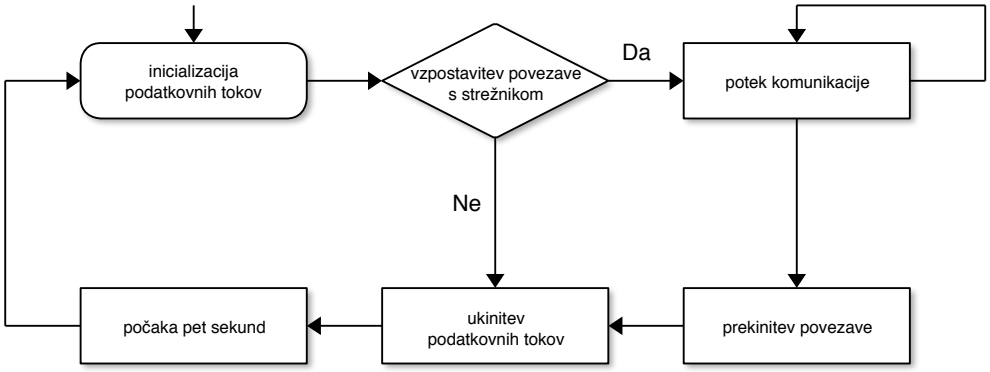

Slika 2.13: Diagram poteka strežnika.

soma preko internetnega omrežja. Po inicializaciji objekta vtič sledi odpiranje povezave. V tem koraku strežnik čaka na povezavo odjemalca. Ko je ta vzpostavljena, sledi izvajanje neskončne `while` zanke, v kateri strežnik čaka na prejem podatkov. Ko prejme podatke, najprej preveri, ali je povezava z odjemalcem še vedno vzpostavljena. Če je ta pogoj resničen, sledi interpretacija prejetih podatkov in izvajanje ustrezne akcije, v nasprotnem primeru se povezava zapre. Program za tem počaka dve sekundi in nato inicializira nov vtič ter čaka na ponovno povezavo odjemalca. Za testiranje delovanja strežnika je bil uporabljen preprost program napisan v jeziku Python, ki se obnaša kot odjemalec. Po vzpostavitvi povezave s strežnikom bere podatke s standardnega vhoda, ki jih nato pošlje strežniku. Oblika sporočil oziroma ukazov za vodenje robota v tem koraku še ni pripravljena, zato se odjemalec in strežnik sporazumevata z uporabo preprostih nizov. Slika 2.13 prikazuje diagram poteka strežnika.

Ko je bilo delovanje strežnika dobro testirano, je sledila izvedba odje-

malca. Za olajšanje dela in za razbremenitev glavnega razreda mobilne aplikacije je bil ustvarjen pomožni razred `CommunicationHelper`, ki vsebuje logiko za inicializacijo in vzpostavitev povezave, pošiljanje in prejemanje sporočil ter prekinjanje povezave. Razvojni paket iOS omogoča zelo preprost način sporazumevanja preko omrežja. Potrebno je ustvariti objekta tipa `NSInputStream` in `NSOutputStream`, ki ju nato preko posebne C funkcije povežemo na strežnikov omrežni naslov. Nadaljno sporazumevanje poteka preko teh dveh objektov. Podatke pošljemo strežniku tako, da sporočilo zapišemo v podatkovni tok objekta `NSOutputStream`, prejemamo pa tako, da beremo iz podatkovnega toka objekta `NSInputStream`. Tako strežnik kot odjemalec omogočata več zaporednih sej v času teka programa. V izogib potrebi po ročni vzpostavitvi povezave s strežnikom, je bila ta funkcionalnost sprogramirana kar v pomožnem razredu. Objekt `CommunicationHelper` se takoj ob inicializaciji poskusi povezati na strežnik. Če strežnik ni na voljo oziroma zavrne komunikacijo, se sproži časovnik `reconnectTimer` tipa `NSTimer`, ki periodično vsakih petih sekund poskuša vzpostaviti povezavo s strežnikom. Ko je povezava uspešno vzpostavljena, se časovnik ukine. Od tega trenutka naprej je komunikacija med strežnikom in odjemalcem omogočena. V primeru prekinitve povezave se podatkovna toka objektov `NSInputStream` in `NSOutputStream` zapreta. Po prekinitvi povezave se ponovno sproži časovnik `reconnectTimer`, ki vsakih pet sekund poskuša odpreti novo povezavo. Na sliki 2.14 je prikazan diagram poteka odjemalca.

Prvoten namen je bila nastavitev strežnika v vlogo dostopne točke, ki bi odjemalcu omogočala vzpostavitev povezave s strežnikom brez uporabe dodatnega omrežja WiFi. Prednost take nastavitve bi bila možnost vodenja robota kjerkoli, neodvisno od prisotnosti brezžičnega omrežja. Nastavitev strežnika v vlogo dostopne točke je odvisna od tega, ali sprejemnik WiFi to omogoča. Uporabljen sprejemnik je takšno delovanje sicer omogočal, vendar je za pravilno delovanje zahteval tudi povezavo z internetnim omrežjem. V nasprotnem primeru je bilo njegovo delovanje zelo nezanesljivo. Zaradi nepravilnega delovanja je bila takšna nastavitev opuščena. Za uspešno sporazume-




Slika 2.14: Diagram poteka odjemalca.

vanje robota z mobilno aplikacijo je bila zato potrebna uporaba brezžičnega omrežja.

Strežnik ob prekinitvi povezave ne ustavi izvajanja programa, ampak v stanju mirovanja čaka na morebitne nove povezave. Tak način delovanja je bil izveden zaradi boljše uporabniške izkušnje. Interakcija z robotom je omogočena le preko mobilne aplikacije, zato mora biti robot vedno pripravljen na vzpostavitev povezave. V nasprotnem primeru bi moral uporabnik ročno pognati program, kar bi zahtevalo povezovanje z računalnikom Raspberry Pi preko protokola SSH ali celo priklop zunanjega zaslona in tipkovnice.

Samodejna vzpostavitev povezave s strani mobilne aplikacije je bila prav tako izvedena zaradi boljse uporabniške izkušnje. Od uporabnika ne zahteva nikakršne interakcije in hkrati odpravi potrebo po dodatnih gumbih na uporabniškem vmesniku, saj mobilna aplikacija sama skrbi za vzpostavitev in prekinitev povezave s strežnikom. Navedeni način delovanja strežnika in odjemalca je precej olajšal proces razvoja in testiranja, saj je odpravil potrebo po ponovnem zagonu programov oziroma potrebo po ročni vzpostavitvi povezave ob zagonu strežnika ali odjemalca.

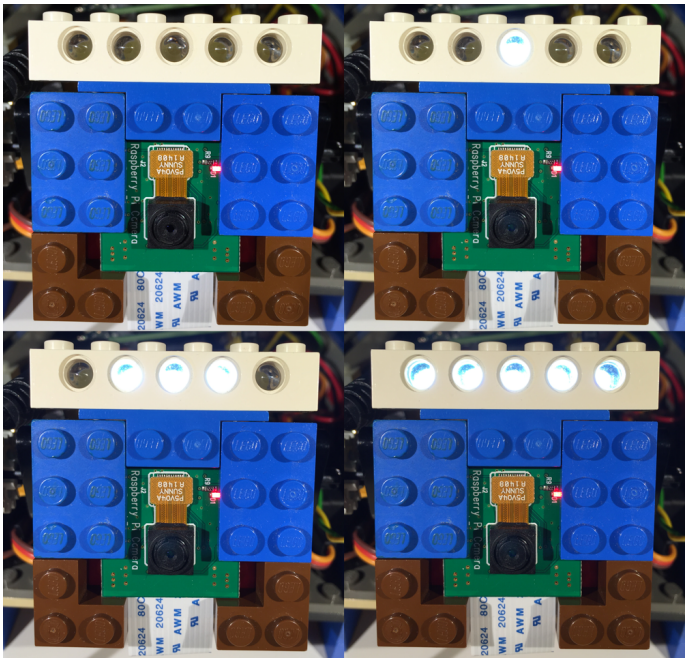
Po izvedbi komunikacije je bil uporabniški vmesnik aplikacije nadgrajen s statusno oznako in oznako napak. Statusna oznaka je po privzetem

rdeče barve in prikazuje besedilo `Not connected`. Tako sporoča uporabniku, da povezava z robotom ni vzpostavljena in njegovo upravljanje ni mogoče. Ob uspešni vzpostavitvi povezave se besedilo oznake nastavi na `Connected`. Njeno ozadje se obarva v zeleno barvo, kar uporabnika obvešča, da je mobilna aplikacija povezana in pripravljena na upravljanje robota. Statusna oznaka se nahaja na zgornjem robu zaslona in je sredinsko poravnana.

Druga nadgradnja uporabniškega vmesnika je bil izpis tipa napake. Ta oznaka se prikaže v primeru napak pri sporazumevanju. Takrat izpiše razlog napake in tako uporabniku pojasni, zakaj povezave ni bilo mogoče vzpostaviti oziroma je bila prekinjena. Do napak v sporazumevanju lahko pride v primeru, ko strežnik zahtevo po povezavi zavrne, ko strežnika ni mogoče najti in ko pride do prekinitve povezave.

Odjemalec in strežnik si med seboj prenašata sporočila v obliki zelo preprostih nizov znakov. Pozamezno sporočilo na primer: `"x y z|"` je sestavljeno iz treh argumentov in navpičnice, ki označuje konec sporočila. Uporaba znaka za konec sporočila olajša strežniku prejemanje in tolmačenje sporočil. Prvi argument sporočila vedno predstavja tip ukaza, druga dva argumenta pa predstavljata parametra ukaza.

Sporočila tipa `0` so namenjena vodenju robota. Prvi parameter predstavlja hitrost levega pogonskega motorja, drugi parameter pa hitrost desnega pogonskega motorja. Sporočila tipa `1` so namenjena nastavljanju naklona kamere. Prvi parameter je v tem primeru vedno `0` in se ne upošteva, drugi parameter pa predstavlja naklon kamere. Sporočila, ki se začnejo s število `2`, vsebujejo parametre za nastavljanje števila prižganih svetlečih diod in njihovo svetilnost. V tem primeru prvi parameter predstavlja svetilnost svetlečih diod, drugi parameter pa stanje svetlečih diod (število prižganih svetlečih diod). Robot ima skupno pet svetlečih diod namenjenih osvetljevanju prostora. Nameščene so v ravni vrsti nad kamero. Drugi argument ne predstalja realnega števila prižganih svetlečih diod, ampak njihovo stanje. Možna stanja so `0`, `1`, `2` in `3`. V stanju `0` so vse svetleče diode izklopljene, v stanju `1` je prižgana samo sredinska svetleča dioda, v stanju `2` so prižgane srednje tri

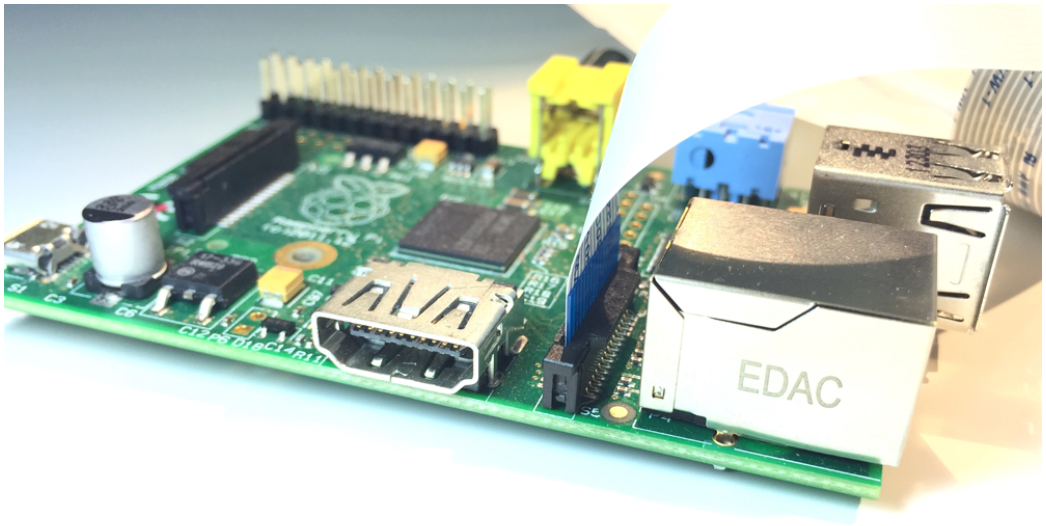


Slika 2.15: Stanja svetlečih diod.

svetleče diode. Stanje `3` označuje da so vse svetleče diode prižgane. Stanja svetlečih diod so prikazana na sliki 2.15.

#### 2.5.2.4 Video prenos

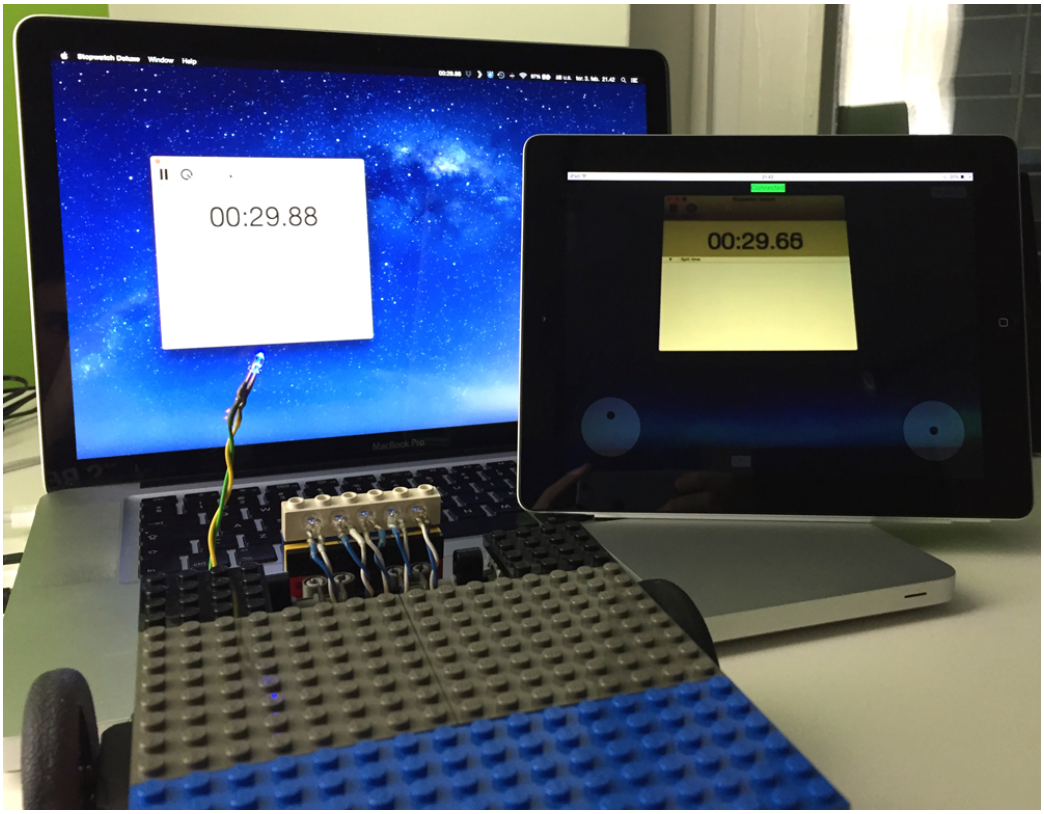
Pred pričetkom izvedbe prenosa pretočnega videa je potrebno kamero najprej povezati z računalnikom Raspberry Pi in znotraj operacijskega sistema omogočiti njeno delovanje. Video kamero je potrebno preko trakovnega kabla povezati na vrata CSI računalnika Raspberry Pi. Ta vrata se nahajajo med priključkoma ethernet in HDMI, kot je vidno na sliki 2.16. Pred testiranjem delovanja kamere je potrebno operacijskemu sistemu omogočiti povezavo s kamero. To storimo z zagonom že znanega ukaza `raspi-config`. Za vključitev kamere je potrebno izbrati možnost `Enable Camera`. Po vkjučitvi kamere zapustimo konfiguracijski zaslon in ponovno zaženemo računalnik. Delovanje kamere lahko preverimo z uporabo ukaza `raspistill`. Z izvajanjem ukaza (`raspistill -o image.jpg`) zajamemo sliko poimenovano `image` formata




Slika 2.16: Vrata CSI, ki služijo povezavi z video kamero.

`jpg`. V trenutni mapi preverimo ali slika obstaja in jo odpremo. Tako se prepričamo, da je kamera pravilno povezana in ustrezno nastavljena. Kamera omogoča tudi snemanje video posnetkov. Z ukazom `raspivid -o video.h264 -t 10000` zajamemo posnetek dolžine 10000 ms, kar ustreza desetim sekundam [30].

Pretočni video omogoča vodenje robota na daljavo, zato je ustrezna izvedba prenosa video podatkov zelo pomembna. Glavna zahteva je omogočanje prenosa video podatkov s kar se da nizko zakasnitvijo, saj visoka zakasnitev onemogoča učinkovito vodenje robota po prostoru. Doseganje nizke zakasnitve se je izkazalo za zelo zahtevno nalogo. Prenos pretočnega videa na računalniku Raspberry Pi omogoča veliko lupinskih programov, vendar je problem večine ravno visoka zakasnitev video prenosa. V povprečju dosegajo približno petsekundno zakasnitev, kar onemogoči učinkovito vodenje robota. Edini program, ki omogoča prenos video podatkov z dovolj nizko zakasnitvijo, je program GSteamer. Slabost tega programa je ta, da zahteva uporabo programa GStreamer tudi na odjemalčevi strani, kar onemogoča možnost predvajanja pretočnega videa na poljubni napravi. Po drugi strani




Slika 2.17: Zakasnitev prenosa pretočnega videa znaša 220 ms.

pa je ravno uporaba namenskega programa tako na strežniku kot na odjemalcu razlog, da lahko dosežemo zakasnitev, manjšo od 500 ms. Ta rezultat v primerjavi z ostalimi programi predstavlja ogromno izboljšavo.

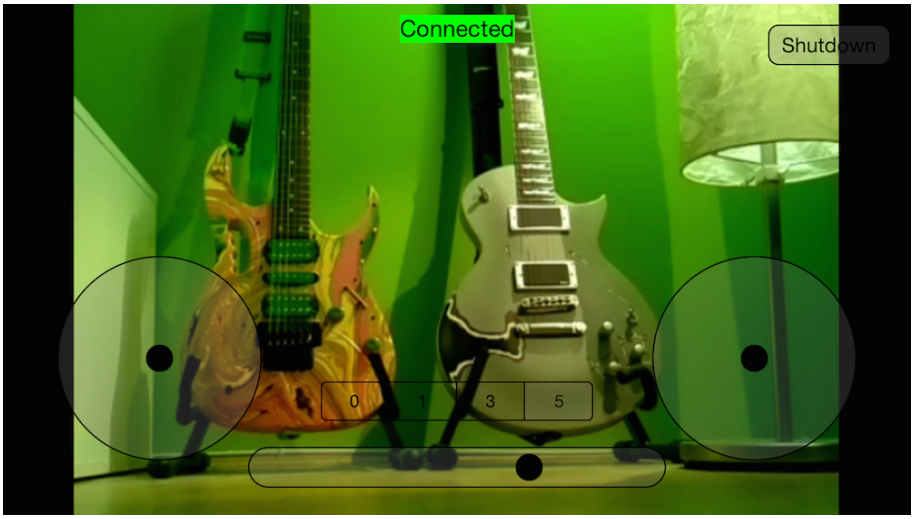
Namestitvi programa GStreamer je sledila izvedba video prenosa v razredu `Robot`. Video prenos se sproži po uspešni vzpostavitvi povezave z odjemalcem, zaustavi pa se ob prekinitvi povezave. Zaradi preprostejše izvedbe se za vzpostavitev in prekinitev video prenosa uporabljajo kar lupinski (Bash) ukazi [31], izvedeni znotraj Python kode. Zagon video prenosa je možen tudi z uporabo Python knjižnice, vendar bi to zahtevalo pisanje dodatne kode, končni učinek pa bi bil popolnoma enak kot pri uporabi lupinskih ukazov. Rezultat uporabe programa GStreamer je bil prenos pretočnega videa ločljivosti 480x320 s 25 sličicami na sekundo in zakasnitvijo manjšo od

500 milisekund (povprečno okrog 250 ms). Na sliki 2.17 je vidna dejanska zakasnitev pretočnega videa (220 ms). Z uporabo internetne povezave preko kabla ethernet (na strežniku in odjemalcu) je mogoče doseči tudi prenos videa v visoki ločljivosti (720p @ 30 fps) z enako zakasnitvijo.

Prikaz pretočnega videa GStreamerjevega strežnika je na platformi iOS mogoč z uporabo razvojnega paketa GStreamer SDK. Zaradi pomanjkanja ustrezne dokumentacije in zelo pomanjkljivih primerov uporabe razvojnega paketa [32] se je prikaz pretočnega videa izkazal za veliko zahtevnejšega kot je sprva izgledal. Največji problem sta predstavljala nastavljanje ustrezne konfiguracije projekta Xcode v stanje, ki je omogočalo prevajanje kode ogrodja GStreamer in nastavljanje parametrov za povezovanje mobilne aplikacije z video strežnikom.

Prvi korak je zahteval dodajanje razvojnega ogrodja GStreamer v projekt aplikacije. Nato je bilo potrebno popraviti nastavitve povezovanja in prevajanja kode projekta, saj se v nasprotnem primeru programska koda ogrodja GStreamer ni prevedla in je tako onemogočila izvajanje programa. Ustrezne nastavitve projekta Xcode za prevajanje kode ogrodja GStreamer so bile ugotovljene šele po večurni raziskavi, pregledu mnogih forumov in nakjučnem poskušanju.

Sledila je dejanska uporaba ogrodja v projektu. Pred prikazovanjem video prenosa se je potrebno najprej povezati na video strežnik robota. To je ločen video strežnik, ki ga upravlja program GStreamer. V pomoč za delo z video strežnikom je bil razvit pomožni razred `StreamHelper`. Koda tega razreda je bil prepisana iz enega od primerov, priloženih ogrodju GStreamer. Iz navedenega razloga se poleg Objective-C kode pojavi tudi običajna C koda. Ta je bila pred uporabo predelana tako, da je ustrezala potrebam razvite mobilne aplikacije. Glavne spremembe so zajemale spreminjanje parametrov za povezovanje na video strežnik robota in prikazovanje pretočnega videa. Nastavitev ustreznih parametrov [31] je enako kot nastavljanje konfiguracije projekta, zahtevala mnogo ur raziskovanja in preizkušanja več različic ogrodja GStreamer. Povezava je bila mogoča šele ob uporabi razvojne različice




Slika 2.18: Končni uporabniški vmesnik na mobilnem telefonu iPhone 6.


Slika 2.19: Končni uporabniški vmesnik na tabličnem računalniku iPad 3.

ogrodja GStreamer [33]. Ta različica ni objavljena na uradni spletni strani. Najdena je bila med objavami na spletnem dnevniku enega od razvijalcev ogrodja [34].

Po uspešni povezavi z video strežnikom je bilo potrebno pretočni video samo še prikazati na uporabniškem vmesniku aplikacije. Za prikaz videa se uporablja zelo preprost razred `VideoFeedView`. Ta deduje od razreda `UIView` in implementira metodo `layerClass`, v kateri vrne objekt tipa `CAEAGLLayer`, ki omogoča izrisovanje vsebine z uporabo orodja OpenGL. Ogrodje GStreamer nato z uporabo tega objekta izriše vsebino pretočnega videa. Objekt `VideoFeedView` je nastavljen kot ozadje mobilne aplikacije. To omogoča spremljanje pretočnega videa in hkratno upravljanje ostalih elementov uporabniškega vmesnika. Povezava z video strežnikom se vzpostavi po uspešni vzpostavitvi povezave s komunikacijskim strežnikom robota in traja, dokler je ta povezava vzpostavljena. Končni vmesnik mobilne aplikacije je viden na slikah 2.18 in 2.19.

### 2.5.3 Združevanje v celoto

#### 2.5.3.1 Ohišje

Ko je bil razvoj robota končan, je sledilo še fizično oblikovanje robota. To je v glavnem zajemalo pripravo ohišja. Računalnik in ostale komponente so bile v procesu razvoja pritrjene na ploščato platformo kock Lego [35]. Uporaba kock Lego se je v fazi razvoja tako dobro obnesla, da se je obdržala tudi pri izgradnji končnega ohišja robota. Pogonska motorja in servo motor za nastavljanje naklona kamere so z vijaki pritrjeni na bloke kock Lego, kar omogoča njihovo pritrditev na ostale kocke Lego. Ohišje temelji na ploščati platformi, na katero so nameščene stranice ohišja in ostale komponente. Z uporabo nekaj dodatnih ploščatih platform je bil izdelan pokrov, ki je na ohišje pritrjen preko pregibnih kock in omogoča dostop do komponent v notranjosti ohišja. Slika 2.20 prikazuje pritrditev pokrova in servo motorjev na ohišje robota. Video kamera je pritrjena na ločeno platformo, ki je preko verižnikov z zobniki povezana s servo motorjem, ki omogoča spreminjanje

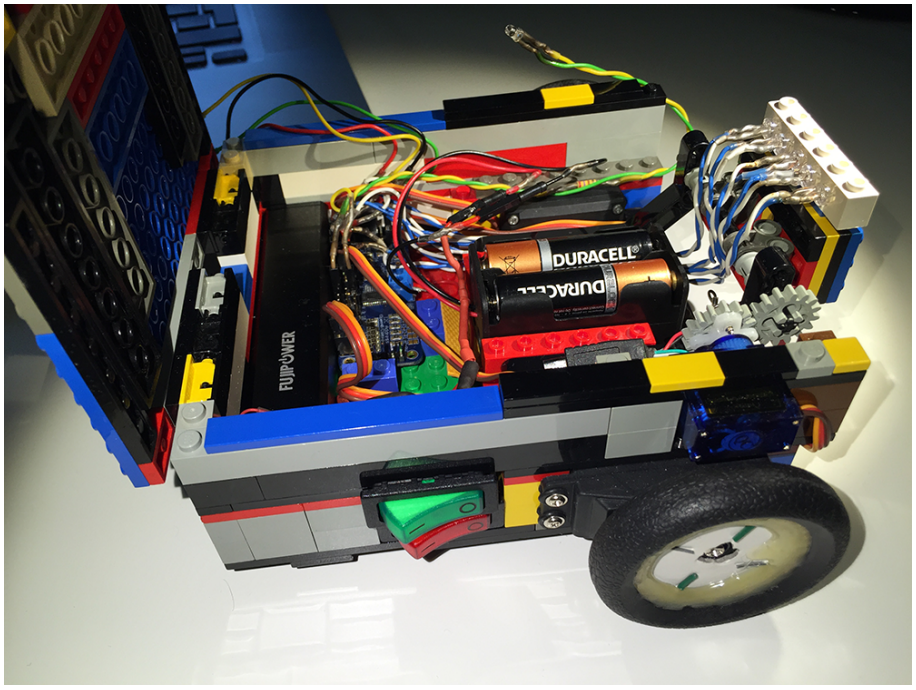
njenega naklona. Ohišje vsebuje dve stikali: prvo omogoči električno napajanje računalnika Raspberry Pi, drugo pa omogoči napajanje servo motorjev. Razporeditev elementov znotraj ohišja robota je bolje vidna na sliki 2.21. Uporaba kock Lego se je izkazala za zelo primerno izbiro, saj omogoča zelo hitro prilagajanje oblike ohišja in notranje razporeditve komponent. Končna oblika razvitega robota je vidna na slikah 2.22 in 2.23.

#### 2.5.3.2 Samodejni zagon

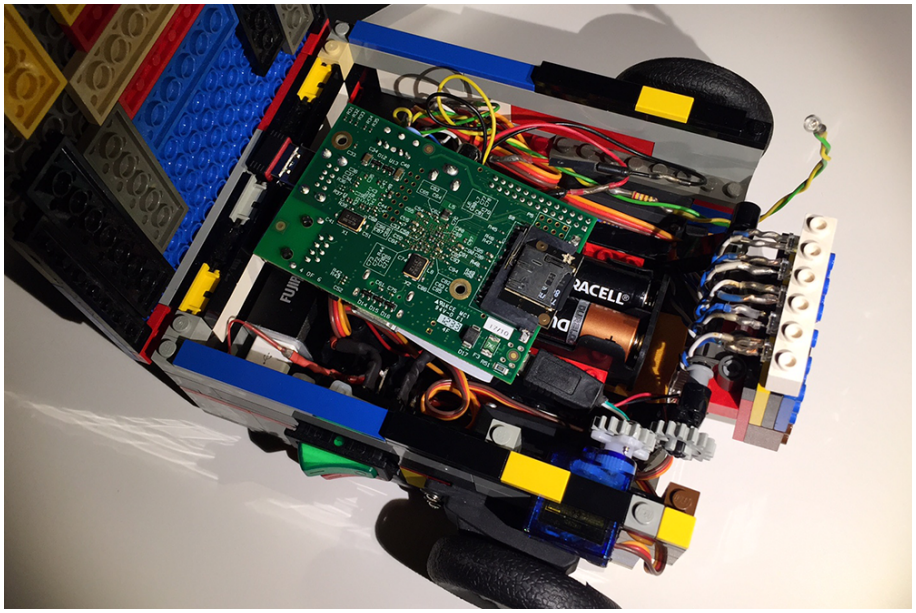
Samodejni zagon je zelo pomembna funkcija robota za doseganje boljše uporabniške izkušnje. V času razvoja je bilo potrebno program vsakokrat zagnati preko povezave SSH. Ta korak bi povzročal težave običajnim, tehnično slabše podkovanim uporabnikom. Poleg tega daje samodejni zagon robotu pridih zakjučenega produkta, ki bi lahko bil ponujen končnemu uporabniku. Python program, ki služi upravljanju delovanja robota, se ob zagonu računalnika Raspberry Pi samodejno zažene preko po meri razvite Bash skripte. Skripto zažene upravljalec opravil Cron ob zagonu operacijskega sistema. To dosežemo z zapisom ukaza `@reboot sh pot_do_skripte` v konfiguracijsko datoteko (`crontab`) programa Cron [36]. Skripta v `while` zanki preverja, ali je računalnik že povezan na internetno omrežje. V primeru, da pogoj ni resničen, počaka nekaj trenutnov in nato ponovno preveri, sicer pa zažene Python program, ki upravlja delovanje robota. Program ob zagonu vklopi tudi modro svetlečo diodo, ki uprabniku signalizira, da je robot v stanju pripravljanosti. Z uporabo mobilne aplikacije se lahko nato nanj povežemo in ga začnemo upravljati.

#### 2.5.3.3 Zaustavitev sistema

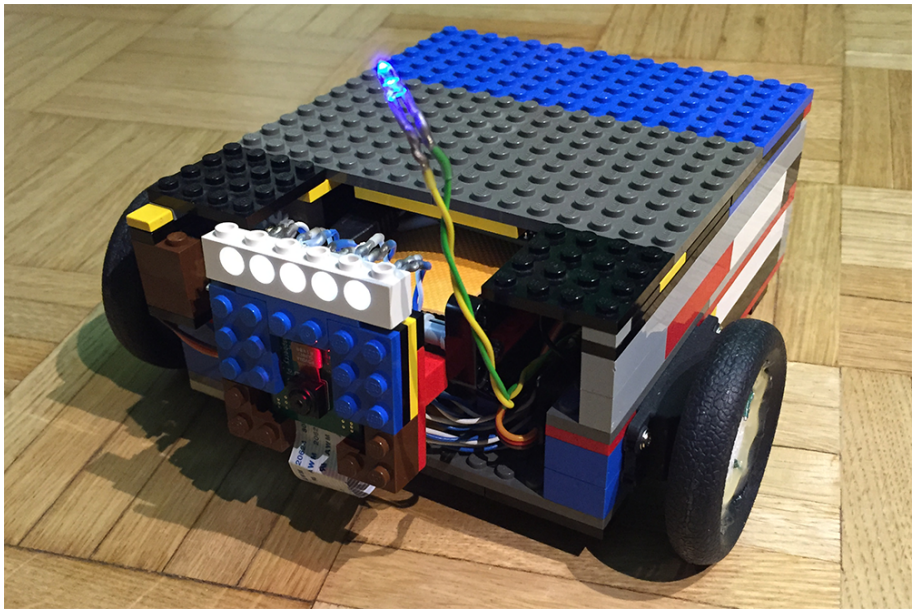
Zaustavitev sistema preko mobilne aplikacije je bila izvedena iz enakega razloga kot samodejni zagon robota. Tako kot ostale računalnike tudi računalnika Raspberry Pi ni priporočljivo ugasniti kar z izkjučitvijo električnega napajanja. Ker je komunikacija uporabnika z robotom mogoča le preko mobilne aplikacije, je potrebno to funkcijo vgraditi v mobilno aplikacijo. V zgornji levi kot uporabniškega vmesnika je bil tako dodan gumb z



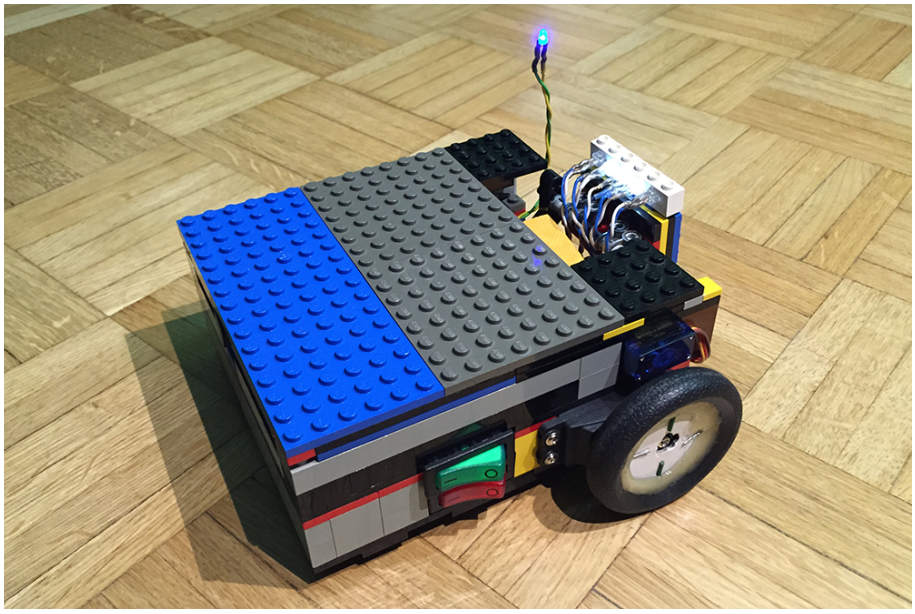

Slika 2.20: Pritrditev pokrova in servo motorjev na bloke kock Lego.



Slika 2.21: Končna razporeditev elementov znotraj ohišja.



Slika 2.22: Končni izgled robota - pogled s prednje strani.



Slika 2.23: Končni izgled robota - pogled s strani.

oznako `Shutdown`, ki je viden le tedaj, ko je vzpostavljena povezava z robotom. Klik na gumb ustvari sporočilo oblike `"3 0 0|"`, ki ga pošlje robotu. Ko ta prejme sporočilo, ustavi video strežnik, prekine povezavo z mobilno aplikacijo in preko lupine izvede ukaz `shutdown -h now`, ki sproži zaustavitev operacijskega sistema. Šele po zaustavitvi sistema lahko prekinemo dovajanje električnega napajanja s premikom obeh napajalnih stikal v izklopljeno stanje.

# Poglavje 3

# Sklepne ugotovitve

Namen diplomskega dela je bil razvoj mobilnega robota, ki temelji na računalniku Raspberry Pi in razvoj programske opreme, ki omogoča njegovo upravljanje. Računalnik Raspberry Pi je bil izbran zaradi želje po preizkusu delovanja računalnika, zaradi možnosti uporabe na različnih področjih ter zaradi radovednosti, kako dobro se njegova uporaba obnese pri projektih, ki se ukvarjajo z razvojem robotov. Ker računalnik Raspberry Pi poganja operacijski sistem Linux, omogoča uporabo visokonivojskih programskih jezikov. Skupaj z množico že razvitih programov ter programskih knjižnic to omogoči izvedbo tudi zahtevnejših robotov. Začrtano idejo je z uporabo računalnika Raspberry Pi mnogo lažje razviti kot z uporabo ostalih mikrokontrolerjev namenjenih preprostim robotom. Ravno zaradi tega je računalnik Raspberry Pi odlična osnova za začetnike, ki se lotijo razvoja prvega robota.

Poudariti je potrebno, da ima računalnik Raspberry Pi na področju upravljanja servo motorjev in svetlečih diod tudi nekaj pomanjkljivosti. Prva pomanjkljivost je ta, da omogoča upravljanje samo ene naprave PWM. Razlog za to je pomanjkanje nožic GPIO, na katere bi lahko povezali dodatne naprave PWM. Druga pomanjkjivost izhaja iz načina delovanja operacijskega sistema. Visokonivojski operacijski sistemi niso primerni za izvajanje časovno tako intenzivnih opravil kot je upravljanje naprav PWM. Čas uporabe procesne enote enakomerno razporejajo med vsemi procesi, kar onemogoči do-

seganje časovne usklajenosti signalov PWM. Te pomanjkljivosti odpravimo z uporabo razširitvenega vezja, ki vsebuje lastni časovnik in opravlja nalogo usklajevanja signalov PWM. Brez uporabe dodatnega razširitvenega vezja računalnik Raspberry Pi ne bi bil primerna platforma za razvoj projektov robotike.

Razviti robot in njegova programska oprema sta bila zasnovana popolnoma po lastni zamisli. Robota je preko mobilne aplikacije mogoče zelo natančno in intuitivno voditi po prostoru. Možno je spremljati njegov pretočni video, kar omogoča vodenje tudi v primerih, ko se robot nahaja zunaj vidnega polja upravljalca. Z uporabo petih svetlečih diod lahko osvetli vidno polje kamere, kar olajša njegovo vodenje po temnih prostorih. Za lažje pregledovanje prostora in zaznavanje ovir nudi tudi možnost spreminjanja naklona video kamere.

Z uporabo zunanje baterije za mobilne telefone kapacitete 2200 mAh in štirih alkalnih baterij doseže čas delovanja do dveh ur. Spremljanje pretočnega videa robota je mogoče pri ločljivosti 480x320 pikslov s 25 sličicami na sekundo pri zakasnitvi manjši od 500 ms (v povprečju 250 ms). Ta rezultat je v primerjavi z ostalimi izvedbami pretočnega videa na računalniku Raspberry Pi več kot desetkrat hitrejši. Ob spremljanju pretočnega videa je robota mogoče zelo natančno voditi po prostoru, zakasnitev je dovolj nizka, da ne povzroča nikakršnih težav. Z uporabo internetne komunikacije preko kabla ethernet je mogoče spremljanje pretočnega videa v visoki ločljivosti (720p @ 30 fps) pri enaki zakasnitvi. Uporaba bolj kvalitetnega sprejemnika WiFi, ki omogoča prenašanje podatkov z višjo hitrostjo, bi izboljšala kvaliteto video posnetka tudi pri uporabi brezžične povezave.

Mobilna aplikacija je bila prav tako po meri razvita za potrebe vodenja robota. Tako uporabniški vmesnik kot delovanje aplikacije je bilo vir lastne zamisli. Namen je bil razviti za uporabnika preprosto aplikacijo, ki omogoča nadzorovanje vseh funkcij robota in spremljanje pretočnega videa. Ob zagonu ne zahteva nikakršne interakcije uporabnika. Z robotom se samodejno poveže in začne prikazovati njegov pretočni video. Uporabniški vmesnik aplikacije

oponaša delovanje igralnega ploščka, kar omogoča zelo intuitiven in učinkovit način vodenja. Robota je mogoče preko mobilne aplikacije tudi ugasniti. S pritiskom na gumb `Shutdown` se sproži prekinitev programa in nato zaustavitev opreacijskega sistema.

Uporaba robota lahko sega na različna področja. Robota takega tipa bi lahko uporabili za pregledovanje in raziskovanje človeku težko dostopnih ali celo nevarnih prostorov. Z uporabo večjih koles, ki bi robotu omogočale lažjo vožnjo in boljše premagovanje ovir, bi robota lahko uporabljali tudi na grobih terenih. V trenutnem stanju je robot namenjem predvsem zabavi in nima praktičnega namena.

Možne izboljšave robota bi lahko vključevale uporabo močnejših servo motorjev in baterij z višjo kapaciteto, ki bi robotu omogočale hitrejše premikanje in bi mu podaljšale čas delovanja. Z uporabo 3D tiskalnika bi lahko izdelali natančno ohišje, na katerega bi laže pritrdili notranje komponente. Tako bi lahko bolje izkoristili notranjo razporeditev komponent in nekoliko pomanjšali ohišje. Dodatna izboljšava bi bila uporaba pospeškomera in giroskopa, s katerima bi robotu omogočili uravnoteženo vožnjo po dveh kolesih in bi tako odpravili potrebo po stabilizacijskem kolesu. Ta nadgradnja bi zahtevala večje spremembe oblike robota, razporeditve komponent ter razporeditve teže robota. Zaradi uporabe kock Lego bi lahko zahtevane spremembe dosegli brez večjih težav. Robota bi bilo mogoče nadgraditi tudi na področju umetne inteligence in računalniškega vida. Dodali bi lahko sposobnost prepoznavanja in samostojnega izmikanja oviram, samostojnega potovanja po prostoru, prepoznavanja objektov in izvajanja raznih akcij, povezanih s prepoznanimi objekti. Možna nadgradnja bi lahko vkjučevala tudi uporabo fotovoltaičnih panelov in komunikacije preko radijske povezave, kar bi omogočilo uporabo robota na oddaljenih lokacijah. To bi seveda zahtevalo nadgradnjo strojne in programske opreme.

Zaradi uporabe računalnika Raspberry Pi in ohišja iz kock Lego omogoča robot mnogo različnih nadgradenj tako na področju strojne kot tudi na področju programske opreme. Delo na tem projektu je bilo zelo praktično

usmerjeno in hkrati zelo zanimiva izkušnja. V posameznih stopnjah razvoja postaja robot vedno bolj napreden, kar razvijalca dodatno spodbuja k delu in nadgrajevanju sposobnosti robota. Razvoj lastno načrtovanega robota je hkrati tudi zelo nagrajujoč proces. Diplomska naloga dokazuje, da je razvoj robota z uporabo računalnika Raspberry Pi in ustrezne strojne opreme mogoč in to z odličnimi rezultati.

***Izvorna koda robota je dostopna na naslovu:***

*https://github.com/crtgregoric/RPi-Robot.git*

***Izvorna koda mobilne aplikacije je dostopna na naslovu:***

*https://github.com/crtgregoric/RPi-Robot-Control.git*

# Literatura


[1] RPi VerifiedPeripherals. Dostopno na:
http://elinux.org/RPi_VerifiedPeripherals

[2] Raspberry Pi. Dostopno na:
http://en.wikipedia.org/wiki/Raspberry_Pi

[3] Adafruit 16-Channel 12-bit PWM/Servo Driver. Dostopno na:
http://www.adafruit.com/product/815

[4] Fundacija Raspberry Pi. Dostopno na:
http://www.raspberrypi.org

[5] Raspberry Pi Camera Module. Dostopno na:
http://www.raspberrypi.org/products/camera-module/

[6] TowerPro SG-5010 - Standard Servo. Dostopno na:
http://www.servodatabase.com/servo/towerpro/sg-5010

[7] TowerPro SG90 - Micro Servo. Dostopno na:
http://www.servodatabase.com/servo/towerpro/sg90

[8] Kartica SD. Dostopno na:
http://en.wikipedia.org/wiki/Secure_Digital

[9] Kingston Micro SD 8 GB Class 4. Dostopno na:
http://www.kingston.com/datasheets/sdc4_en.pdf

[10] Transcend Micro SD 8 GB Class 10. Dostopno na:
http://www.transcend-info.com/Products/No-320

[11] EDUP Ultra-Mini Nano Network Adapter. Dostopno na:
http://www.edupdriver.com/edup-ep-n8508-802-11n-wireless-lan-usb-ultra-mini-nano-network-adapter/

[12] How to portably power your Raspberry Pi with a battery. Dostopno na:
http://www.thefruitycomputer.com/forums/tutorials/article/17-how-to-portably-power-your-raspberry-pi-with-a-battery/

[13] Running a Raspberry Pi from 6 AA Batteries. Dostopno na:
http://www.raspberrypi-spy.co.uk/2013/02/running-a-raspberry-pi-from-6-aa-batteries/

[14] iOS. Dostopno na:
https://developer.apple.com/technologies/ios/

[15] Xcode. Dostopno na:
https://developer.apple.com/xcode/

[16] Adafruit Raspberry Pi Educational Linux Distro. Dostopno na:
https://learn.adafruit.com/adafruit-raspberry-pi-educational-linux-distro/occidentalis-v0-dot-2

[17] Raspbian. Dostopno na:
http://www.raspbian.org

[18] Debian. Dostopno na:
https://www.debian.org

[19] GStreamer. Dostopno na:
http://gstreamer.freedesktop.org

[20] Pulse-width modulation. Dostopno na:
http://arduino.cc/en/Tutorial/PWM

[21] A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators. Dostopno na: http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html

[22] RPi Easy SD Card Setup. Dostopno na: http://elinux.org/RPi_Easy_SD_Card_Setup

[23] Adafruit 16 Channel Servo Driver with Raspberry Pi. Dostopno na: https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi

[24] Adafruit-Raspberry-Pi-Python-Code. Dostopno na: https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code

[25] PyCharm. Dostopno na: https://www.jetbrains.com/pycharm/

[26] SSH File Transfer Protocol. Dostopno na: http://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol

[27] How to Share Files between Mac and Linux the Easy Way. Dostopno na: http://blog.austinseraphin.com/2011/08/07/how-to-share-files-between-mac-and-linux-the-easy-way/

[28] Bonjour/Zeroconf. Dostopno na: http://en.wikipedia.org/wiki/Bonjour_(software)

[29] Low-level networking interface. Dostopno na: https://docs.python.org/2/library/socket.html

[30] How To Install / Use The Raspberry Pi Camera. Dostopno na: http://thepihut.com/blogs/raspberry-pi-tutorials/16021420-how-to-install-use-the-raspberry-pi-camera

[31] Raspberry Pi camera board – Gstreamer. Dostopno na: http://pi.gbaman.info/?p=150

[32] iOS tutorials - GStreamer SDK documentation. Dostopno na: http://docs.gstreamer.com/display/GstSDK/iOS+tutorials

[33] GStreamer iOS SDK v1.4.4. Dostopno na: http://gstreamer.freedesktop.org/pkg/ios/1.4.4/

[34] GStreamer 1.0 examples for iOS, Android and in general. Dostopno na: https://coaxion.net/blog/2013/10/gstreamer-1-0-examples-for-ios-android-and-in-general/

[35] Kocke Lego. Dostopno na: http://www.lego.com/

[36] Launch Python script on startup. Dostopno na: http://www.instructables.com/id/Raspberry-Pi-Launch-Python-script-on-startup/step4/Add-to-your-crontab/